C# Programming Examples

This section covers the list of topics for C# programming examples. These C# examples cover a wide range of programming areas in Computer Science. Every example program includes the description of the program, C# code as well as output of the program. All examples are compiled and tested on Visual Studio. These examples can be as simple and basic as “Hello World” program to extremely tough and advanced C# programs. So, they are suitable for any user (dummies, beginners or advanced users).  
Here is the listing of C# programming Topics:

A. C# Basic Programming Examples

The following collection contains various C# programs on mathematical operations, date and year formats, swapping and bitwise operations, interface, looping and numerical operations, access specifiers and various other programs on unboxing operation and sealed classes.

#### 1. C# Examples on Fundamental Mathematical Operations

The following section deals with fundamental mathematical operations. Numbers ending with 0,2,4,6,8 are called even numbers and numbers ending with 1,3,5,7,9 are called odd numbers. A Binary Triangle is the triangle formed by a series of 0’s and 1’s. So, the C# Programs in the following section checks for even or odd properties of a number, interchanges the values of 2 numbers which is done using the process of swapping, displays the sum of digits in a number, displays the reverse of a number, prints a binary triangle and counts the number of 1’s in a given number which is input to the program.

C# Program to Check whether the Entered Number is Even or Odd

This C# Program checks if a given integer is Odd or Even. Here if a given number is divisible by 2 with the remainder 0 then the number is an Even number. If the number is not divisible by 2 then that number will be an Odd number.

Here is source code of the C# program which checks a given integer is odd or even. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Check whether the Entered Number is Even or Odd*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
9. **namespace** check1
10. {
11. **class** Program
12. {
13. **static** **void** Main(**string**[] args)
14. {
15. **int** i;
16. Console.Write("Enter a Number : ");
17. i = **int**.Parse(Console.ReadLine());
18. **if** (i % 2 == 0)
19. {
20. Console.Write("Entered Number is an Even Number");
21. Console.Read();
22. }
23. **else**
24. {
25. Console.Write("Entered Number is an Odd Number");
26. Console.Read();
27. }
28. }
29. }
30. }

Here is the output of the C# Program:

Enter a Number : 25

Entered Number is an Odd Number

C# Program to Swap 2 Numbers

This C# Program Swaps 2 Numbers.It obtains two numbers from the user and swaps the numbers using a temporary variable.

Here is source code of the C# program that swaps two numbers. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Swap two Numbers*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
8. **namespace** Program
9. {
10. **class** Program
11. {
12. **static** **void** Main(**string**[] args)
13. {
14. **int** num1, num2, temp;
15. Console.Write("**\n**Enter the First Number : ");
16. num1 = **int**.Parse(Console.ReadLine());
17. Console.Write("**\n**Enter the Second Number : ");
18. num2 = **int**.Parse(Console.ReadLine());
19. temp = num1;
20. num1 = num2;
21. num2 = temp;
22. Console.Write("**\n**After Swapping : ");
23. Console.Write("**\n**First Number : "+num1);
24. Console.Write("**\n**Second Number : "+num2);
25. Console.Read();
26. }
27. }
28. }

Here is the output of the C# Program:

Enter the First Number : 23

Enter the Second Number : 25

After Swapping :

First Number : 25

Second Number : 23

C# Program to Get a Number and Display the Sum of the Digits

This C# Program Gets a Number and Display the Sum of the Digits.The digit sum of a given integer is the sum of all its digits.

Here is source code of the C# Program to Get a Number and Display the Sum of the Digits . The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Get a Number and Display the Sum of the Digits*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
9. **namespace** Program
10. {
11. **class** Program
12. {
13. **static** **void** Main(**string**[] args)
14. {
15. **int** num, sum = 0, r;
16. Console.WriteLine("Enter a Number : ");
17. num = **int**.Parse(Console.ReadLine());
18. **while** (num != 0)
19. {
20. r = num % 10;
21. num = num / 10;
22. sum = sum + r;
23. }
24. Console.WriteLine("Sum of Digits of the Number : "+sum);
25. Console.ReadLine();
27. }
28. }
29. }

advertisements

Here is the output of the C# Program:

Enter a Number : 123

Sum of Digits of the Number : 6

C# Program to Get a Number and Display the Number with its Reverse

This C# Program Gets a Number and Display the Number with its Reverse. Here we obtain a number from the user and display the digits in the reverse order.

Here is source code of the C# Program to Get a Number and Display the Number with its Reverse. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Get a Number and Display the Number with its Reverse*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
9. **namespace** Program
10. {
11. **class** Program
12. {
13. **static** **void** Main(**string**[] args)
14. {
15. **int** num, reverse = 0;
16. Console.WriteLine("Enter a Number : ");
17. num = **int**.Parse(Console.ReadLine());
18. **while** (num != 0)
19. {
20. reverse = reverse \* 10;
21. reverse = reverse + num % 10;
22. num = num / 10;
23. }
24. Console.WriteLine("Reverse of Entered Number is : "+reverse);
25. Console.ReadLine();
27. }
28. }
29. }

advertisements

Here is the output of the C# Program:

Enter a Number : 123

Reverse of Entered Number : 321

C# Program to Print a BinaryTriangle

This C# Program Prints a Binary Triangle. Binary Triangle is a Triangle formed with 1’s and 0’s.Number of rows in the binary triangle is obtained from the user.

Here is source code of the C# Program to Print a Binary Triangle. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Print a BinaryTriangle*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
9. **namespace** Program
10. {
11. **class** Program
12. {
13. **public** **static** **void** Main(**String**[] args)
14. {
15. **int** p, lastInt = 0, input;
16. Console.WriteLine("Enter the Number of Rows : ");
17. input = **int**.Parse(Console.ReadLine());
18. **for** (**int** i = 1; i <= input; i++)
19. {
20. **for** (p = 1; p <= i; p++)
21. {
22. **if** (lastInt == 1)
23. {
24. Console.Write("0");
25. lastInt = 0;
26. }
27. **else** **if** (lastInt == 0)
28. {
29. Console.Write("1");
30. lastInt = 1;
31. }
32. } Console.Write("**\n**");
33. }
34. Console.ReadLine();
35. }
36. }
37. }

advertisements

Here is the output of the C# Program:

Enter the Number of Rows : 5

1

01

010

1010

10101

C# Program to Count the Number of 1’s in the Entered Number

This C# Program Counts the Number of 1’s in the Entered Number. Here the array of numbers are obtained with its limit and number of 1’s in it is counted and displayed.

Here is source code of the C# Program to Count the Number of 1’s in the Entered Number. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Count the Number of 1's in the Entered Number*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
9. **namespace** ConsoleApplication16
10. {
11. **class** Program
12. {
13. **static** **void** Main(**string**[] args)
14. {
15. **int** m, count = 0;
16. Console.WriteLine("Enter the Limit : ");
17. m = **int**.Parse(Console.ReadLine());
18. **int**[] a = new **int**[m];
19. Console.WriteLine("Enter the Numbers :");
20. **for** (**int** i = 0; i < m; i++)
21. {
22. a[i] = Convert.ToInt32(Console.ReadLine());
23. }
24. **foreach** (**int** o **in** a)
25. {
26. **if** (o == 1)
27. {
28. count++;
29. }
30. }
31. Console.WriteLine("Number of 1's in the Entered Number : ");
32. Console.WriteLine(count);
33. Console.ReadLine();
34. }
35. }
36. }

advertisements

Here is the output of the C# Program:

Enter the Limit : 5

Enter the Numbers :

1

2

1

4

1

Number of 1's in the Entered Number : 3

#### 2. C# Examples on Date Formats and Years

If the given year is divisible by 4, then it is said to be a Leap Year. A Leap Year has 366 days. The C# Programs in the given section check if the entered year is a leap year or not, they also display various date formats. The different date formats are dd/mm/yyyy, mm/dd/yyyy, dd/mmm/yyyy and so on and compares the two entered dates. It also displays day-to-day ATM transactions and generates random numbers. A random number is generated by selecting a specified distribution that tries to return values in the range of values generated by that distribution.

C# Program to Check Whether the Entered Year is a Leap Year or Not

This C# Program Checks Whether the Entered Year is a Leap Year or Not.When A year is divided by 4. If remainder becomes 0 then the year is called a leap year..

Here is source code of the C# Program to Check Whether the Entered Year is a Leap Year or Not. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Check Whether the Entered Year is a Leap Year or Not*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
9. **namespace** Program
10. {
11. **class** leapyear
12. {
13. **static** **void** Main(**string**[] args)
14. {
15. leapyear obj = new leapyear();
16. obj.readdata();
17. obj.leap();
18. }
19. **int** y;
20. **public** **void** readdata()
21. {
22. Console.WriteLine("Enter the Year in Four Digits : ");
23. y = Convert.ToInt32(Console.ReadLine());
24. }
25. **public** **void** leap()
26. {
27. **if** ((y % 4 == 0 && y % 100 != 0) || (y % 400 == 0))
28. {
29. Console.WriteLine("{0} is a Leap Year", y);
30. }
31. **else**
32. {
33. Console.WriteLine("{0} is not a Leap Year", y);
34. }
35. Console.ReadLine();
36. }
37. }
38. }

advertisements

Here is the output of the C# Program:

Enter the Year in Four Digits : 1004

1004 is a Leap Year

C# Program to Display the Date in Various Formats

This CC# Program to Display the Date in Various Formats . Here the Date is Displayed in various Formats.

Here is source code of the C# Program to Display the Date in Various Formats . The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Display the Date in Various Formats*
3. *\*/*
4. **using** System;
5. **namespace** DateAndTime
6. {
7. **class** Program
8. {
9. **static** **int** Main()
10. {
11. DateTime date = new DateTime(2013,6, 23);
12. Console.WriteLine("Some Date Formats : ");
13. Console.WriteLine("Date and Time: {0}", date);
14. Console.WriteLine(date.ToString("yyyy-MM-dd"));
15. Console.WriteLine(date.ToString("dd-MMM-yy"));
16. Console.WriteLine(date.ToString("M/d/yyyy"));
17. Console.WriteLine(date.ToString("M/d/yy"));
18. Console.WriteLine(date.ToString("MM/dd/yyyy"));
19. Console.WriteLine(date.ToString("MM/dd/yy"));
20. Console.WriteLine(date.ToString("yy/MM/dd"));
21. Console.Read();
22. **return** 0;
23. }
24. }
25. }

advertisements

Here is the output of the C# Program:

Some Date Formats :

Date and Time : 6/23/2013 12:00:00 AM

2013-06-23

23-Jun-13

6/23/2013

6/23/13

06/23/2013

06/23/13

13/06/23

C# Program to Compare Two Dates

This C# Program to Compare Two Dates. Here two dates are compared and the date which occurs first is displayed.

Here is source code of the C# Program to Compare Two Dates. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Campare Two Dates*
3. *\*/*
4. **using** System;
5. **namespace** DateAndTime
6. {
7. **class** Program
8. {
9. **static** **int** Main()
10. {
11. DateTime sd = new DateTime(2010, 10, 12);
12. Console.WriteLine("Starting Date : {0}", sd);
13. DateTime ed = sd.AddDays(10);
14. Console.WriteLine("Ending Date : {0}", ed);
15. **if** (sd < ed)
16. Console.WriteLine("{0} Occurs Before {1}", sd, ed);
17. Console.Read();
18. **return** 0;
20. }
21. }
22. }

advertisements

Here is the output of the C# Program:

Starting Date : 10/11/2010 12:00:00 AM

Ending Date : 10/21/2010 12:00:00 AM

10/11/2010 12:00:00 Am Occurs Before 10/21/2010 12:00:00 AM

C# Program to Display the ATM Transaction

This C# Program Displays the ATM Transaction. Here The types of ATM transaction are  
1) Balance checking  
2) Cash withdrawal  
3) Cash deposition.  
You can opt any of the above transaction according to your need of transaction.

Here is source code of the C# Program to Display the ATM Transaction. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Display the ATM Transaction*
3. *\*/*
4. **using** System;
5. **class** program
6. {
7. **public** **static** **void** Main()
8. {
10. **int** amount = 1000, deposit, withdraw;
11. **int** choice, pin = 0, x = 0;
12. Console.WriteLine("Enter Your Pin Number ");
13. pin = **int**.Parse(Console.ReadLine());
14. **while** (**true**)
15. {
16. Console.WriteLine("\*\*\*\*\*\*\*\*Welcome to ATM Service\*\*\*\*\*\*\*\*\*\*\*\*\*\***\n**");
17. Console.WriteLine("1. Check Balance**\n**");
18. Console.WriteLine("2. Withdraw Cash**\n**");
19. Console.WriteLine("3. Deposit Cash**\n**");
20. Console.WriteLine("4. Quit**\n**");
21. Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***\n\n**");
22. Console.WriteLine("Enter your choice: ");
23. choice = **int**.Parse(Console.ReadLine());
24. **switch** (choice)
25. {
26. **case** 1:
27. Console.WriteLine("**\n** YOUR BALANCE IN Rs : {0} ", amount);
28. **break**;
29. **case** 2:
30. Console.WriteLine("**\n** ENTER THE AMOUNT TO WITHDRAW: ");
31. withdraw = **int**.Parse(Console.ReadLine());
32. **if** (withdraw % 100 != 0)
33. {
34. Console.WriteLine("**\n** PLEASE ENTER THE AMOUNT IN MULTIPLES OF 100");
35. }
36. **else** **if** (withdraw > (amount - 500))
37. {
38. Console.WriteLine("**\n** INSUFFICENT BALANCE");
39. }
40. **else**
41. {
42. amount = amount - withdraw;
43. Console.WriteLine("**\n\n** PLEASE COLLECT CASH");
44. Console.WriteLine("**\n** YOUR CURRENT BALANCE IS {0}", amount);
45. }
46. **break**;
47. **case** 3:
48. Console.WriteLine("**\n** ENTER THE AMOUNT TO DEPOSIT");
49. deposit = **int**.Parse(Console.ReadLine());
50. amount = amount + deposit;
51. Console.WriteLine("YOUR BALANCE IS {0}", amount);
52. **break**;
53. **case** 4:
54. Console.WriteLine("**\n** THANK U USING ATM");
55. **break**;
56. }
57. }
58. Console.WriteLine("**\n\n** THANKS FOR USING OUT ATM SERVICE");
59. }
60. }

advertisements

Here is the output of the C# Program:

Enter Your Pin Number

123

\*\*\*\*\*\*\*\*Welcome to ATM Service\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. Check Balance

2. Withdraw Cash

3. Deposit Cash

4. Quit

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Enter your choice:

1

YOUR BALANCE IN Rs : 1000

C# Program to Generate Random Numbers

This C# Program Generates Random Numbers. Here random numbers are generated using the Random class and the next() in it.

Here is source code of the C# Program to Generate Random Numbers. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Generate Random Numbers*
3. *\*/*
4. **using** System;
5. **class** Program
6. {
7. **static** **void** Main()
8. {
9. Console.WriteLine("Some Random Numbers that are generated are : ");
10. **for** (**int** i = 1; i < 10; i++)
11. {
12. Randfunc();
13. }
14. }
15. **static** Random r = new Random();
16. **static** **void** Randfunc()
17. {
18. **int** n = r.Next();
19. Console.WriteLine(n);
20. Console.ReadLine();
21. }
22. }

advertisements

Here is the output of the C# Program:

Some Random Numbers that are generated are :

1234567

8754352

9864930

8352048

1920472

2846104

7649207

4928756

9261746

#### 3. C# Examples on Bitwise and Swapping Operations

The bitwise exclusive OR operator (^) compares each bit of its first operand to the corresponding bit of its second operand. If one bit is 0 and the other bit is 1, the corresponding result bit is set to 1. otherwise, the corresponding result bit is set to 0. The following C# Programs swap the contents of two numbers using bitwise XOR Operation, it finds the number of times the word ‘the’ (frequency) appears in a given sentence and accept the height of a person & categorizes it as Taller, Dwarf & Average.

C Program to Accept the Height of a Person & Categorize as Taller, Dwarf & Average

This C Program to Accept the Height of a Person & Categorize as Taller, Dwarf & Average. Here The program accepts height of a person in centimeters. Then categorize based on the height. If height is less than 150 centimeter, then the person is dwarf and if the height is between 151 to 165 then it is categorized as average and if the height is between 165 to 175 then it is categorized as taller.

Here is source code of the C Program to Accept the Height of a Person & Categorize as Taller, Dwarf & Average. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C Program to Accept the Height of a Person & Categorize as*
3. *\* Taller, Dwarf & Average*
4. *\*/*
5. **using** System;
6. **class** program
7. {
8. **public** **static** **void** Main()
9. {
10. **float** height;
11. Console.WriteLine("Enter the Height (in centimetres) **\n**");
12. height = **int**.Parse(Console.ReadLine());
13. **if** (height < 150.0)
14. Console.WriteLine("Dwarf **\n**");
15. **else** **if** ((height >= 150.0) && (height <= 165.0))
16. Console.WriteLine(" Average Height **\n**");
17. **else** **if** ((height >= 165.0) && (height <= 195.0))
18. Console.WriteLine("Taller **\n**");
19. **else**
20. Console.WriteLine("Abnormal height **\n**");
21. }
22. }

advertisements

Here is the output of the C# Program:

Enter the Height (in centimetres)

165

Average Height

C# Program to Find the Frequency of the Word ʺtheʺ in a given Sentence

This C# Program Finds the Frequency of the Word ʺtheʺ in a given Sentence. Here the frequency of ‘the’ in the given string is found.

Here is source code of the C# Program to Find the Frequency of the Word ʺtheʺ in a given Sentence. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Find the Frequency of the Word ʺtheʺ in a given Sentence*
3. *\*/*
4. **using** System;
5. **class** program
6. {
7. **public** **static** **void** Main()
8. {
9. **string** s1;
10. Console.WriteLine("Enter the String : ");
11. s1 = Console.ReadLine();
12. Console.WriteLine(counting.CountStringOccurrences(s1, "the"));
13. Console.ReadLine();
14. }
15. }
16. **public** **static** **class** counting
17. {
18. **public** **static** **int** CountStringOccurrences(**string** text, **string** pattern)
19. {
20. **int** count = 0;
21. **int** i = 0;
22. **while** ((i = text.IndexOf(pattern, i)) != -1)
23. {
24. i += pattern.Length;
25. count++;
26. }
27. **return** count;
28. }
29. }

advertisements

Here is the output of the C# Program:

Enter the String :

we only loop once over the source, which reduces the cost of the method.

3

C# Program to Swap the Contents of two Numbers using Bitwise XOR Operation

This C# Program to Swap the Contents of two Numbers using Bitwise XOR Operation.

Here is source code of the C# Program to Swap the Contents of two Numbers using Bitwise XOR Operation. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Swap the Contents of two Numbers using Bitwise XOR Operation*
3. *\*/*
4. **using** System;
5. **class** program
6. {
7. **public** **static** **void** Main()
8. {
9. **int** i, k;
10. Console.WriteLine("Enter two integers **\n**");
11. i = **int**.Parse(Console.ReadLine());
12. k = **int**.Parse(Console.ReadLine());
13. Console.WriteLine("**\n** Before swapping i= {0} and k = {1}", i, k);
14. i = i ^ k;
15. k = i ^ k;
16. i = i ^ k;
17. Console.WriteLine("**\n** After swapping i= {0} and k = {1}", i, k);
18. Console.ReadLine();
19. }
20. }

advertisements

Here is the output of the C# Program:

Enter two integers

23

34

Before swapping i= 23 and k = 34

After swapping i= 34 and k = 23

#### 4. C# Examples on Access Specifiers

Access Specifiers determine whether a field or method in a class, can be used or invoked by another method in another class or sub-class. Public, Private, Protected and Default are the four access specifiers. Loop within another loop is called nested loop. The left-shift operator (<<), which moves the bits of shift\_expression to the left. The following C# programs illustrate the use of access specifiers, left shift operators, prints a diamond using nested loop, reads a grade & displays the equivalent description and finds the greatest among 2 numbers.

C# Program to Illustrate the Use of Access Specifiers

This C# Program Illustrates the Use of Access Specifiers. Here the four types of access specifiers are explained by assigning values.

Here is source code of the C# Program to Illustrate the Use of Access Specifiers. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Illustrate the Use of Access Specifiers*
3. *\*/*
4. **using** System;
5. **namespace** accessspecifier
6. {
7. **class** Program
8. {
9. **static** **void** Main(**string**[] args)
10. {
11. two B = new two();
12. B.show();
13. }
14. }
15. **class** one
16. {
17. **private** **int** x;
18. **protected** **int** y;
19. **internal** **int** z;
20. **public** **int** a;
21. **protected** **internal** **int** b;
22. }
23. **class** two : one
24. {
25. **public** **void** show()
26. {
27. Console.WriteLine("Values are : ");
28. *//x=10;*
29. y = 20;
30. z = 30;
31. a = 40;
32. b = 50;
33. *// Console.WriteLine(+x); // Error x is not accessible*
34. Console.WriteLine(y);
35. Console.WriteLine(z);
36. Console.WriteLine(a);
37. Console.WriteLine(b);
38. Console.ReadLine();
40. }
41. }
42. }

advertisements

Here is the output of the C# Program:

Values are :

20

30

40

50

C# Program to Print a Diamond Using Nested Loop

This C# Program Prints a Diamond Using Nested Loop.

Here is source code of the C# Program to Print a Diamond Using Nested Loop. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Print the Sum of all the Multiples of 3 and 5*
3. *\*/*
4. **using** System;
5. **class** program
6. {
7. **public** **static** **void** Main()
8. {
9. **int** number, i, k, count = 1;
10. Console.Write("Enter number of rows**\n**");
11. number = **int**.Parse(Console.ReadLine());
12. count = number - 1;
13. **for** (k = 1; k <= number; k++)
14. {
15. **for** (i = 1; i <= count; i++)
16. Console.Write(" ");
17. count--;
18. **for** (i = 1; i <= 2 \* k - 1; i++)
19. Console.Write("\*");
20. Console.WriteLine();
21. }
22. count = 1;
23. **for** (k = 1; k <= number - 1; k++)
24. {
25. **for** (i = 1; i <= count; i++)
26. Console.Write(" ");
27. count++;
28. **for** (i = 1; i <= 2 \* (number - k) - 1; i++)
29. Console.Write("\*");
30. Console.WriteLine();
31. }
32. Console.ReadLine();
33. }
34. }

advertisements

Here is the output of the C# Program:

Enter number of rows

3

\*

\*\*\*

\*\*\*\*\*

\*\*\*

\*

C# Program to Illustrate LeftShift Operations

This C# Program Illustrates LeftShift Operations. The shift operators allow programmers to adjust an integer by shifting all of its bits to the left or the right.

Here is source code of the C# Program to Illustrate LeftShift Operations. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Illustrate LeftShift Operations*
3. *\*/*
4. **using** System;
5. **class** sample
6. {
7. **public** **static** **void** Main()
8. {
9. **int** x = 1024 \* 1024 \* 1024;
10. **uint** p = 1024 \* 1024 \* 1024;
11. **int** y = -42;
12. Console.WriteLine("LEFT SHIFT OPERATIONS :");
13. Console.WriteLine("{0},{1},{2}", x, x \* 2, x << 1);
14. Console.WriteLine("{0},{1},{2}", p, p \* 2, p << 1);
15. Console.WriteLine("{0},{1},{2}", x, x \* 4, x << 2);
16. Console.WriteLine("{0},{1},{2}", p, p \* 4, p << 2);
17. Console.WriteLine("{0},{1},{2}", y, y \* 1024 \* 1024 \* 64, x << 26);
18. Console.ReadLine();
19. }
20. }

advertisements

Here is the output of the C# Program:

LEFT SHIFT OPERATIONS :

1073741824,-2147483648,-2147483648

1073741824,2147483648,2147483648

1073741824,0,0

1073741824,0,0

-42,1476395008,0

C# Program to Check whether the given Integer has an Alternate Pattern

This C# Program Checks whether the given Integer has an Alternate Pattern.

Here is source code of the C# Program to Check whether the given Integer has an Alternate Pattern. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Check whether the given Integer has an*
3. *\* Alternate Pattern*
4. *\*/*
5. **using** System;
6. **using** System.Collections.Generic;
7. **using** System.Linq;
8. **using** System.Text;
9. **class** program
10. {
11. **public** **static** **void** Main()
12. {
13. **int** num, x, y, count = 0;
14. Console.WriteLine("Enter the Number:");
15. num = **int**.Parse(Console.ReadLine());
16. x = num << 1;
17. y = x ^ num;
18. y = y + 1;
19. **while** ((y / 2) != 0)
20. {
21. **if** (y % 2 != 0)
22. {
23. count++;
24. **break**;
25. }
26. **else**
27. {
28. y = y / 2;
29. }
30. }
31. **if** (count == 1)
32. {
33. Console.WriteLine("false");
34. }
35. **else**
36. {
37. Console.WriteLine("true");
38. }
39. Console.Read();
40. }
41. }

advertisements

Here is the output of the C# Program:

Enter the Number: 100

false

C# Program to Read a Grade & Display the Equivalent Description

This C# Program Reads a Grade & Display the Equivalent Description. Here If grade is S, it prints super, if grade is A, it prints very good, if grade is B, it prints fair, if grade is Y, it prints absent, if grade is F, it prints fail.

Here is source code of the C# Program to Read a Grade & Display the Equivalent Description. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Read a Grade & Display the Equivalent Description*
3. *\*/*
4. **using** System;
5. **using** System.IO;
6. **class** program
7. {
8. **public** **static** **void** Main()
9. {
10. **char** grade;
11. Console.WriteLine("Enter the Grade in UpperCase **\n**");
12. grade = Convert.ToChar(Console.ReadLine());
13. **switch** (grade)
14. {
15. **case** 'S':
16. Console.WriteLine(" SUPER");
17. **break**;
18. **case** 'A':
19. Console.WriteLine(" VERY GOOD");
20. **break**;
21. **case** 'B':
22. Console.WriteLine(" FAIR");
23. **break**;
24. **case** 'Y':
25. Console.WriteLine(" ABSENT");
26. **break**;
27. **case** 'F':
28. Console.WriteLine(" FAIL");
29. **break**;
30. **default**:
31. Console.WriteLine("ERROR IN GRADE **\n**");
32. **break**;
33. Console.ReadLine();
34. }
35. }
36. }

advertisements

Here is the output of the C# Program:

Enter the Grade in UpperCase

A

VERY GOOD

C# Program to Find Greatest among 2 numbers

This C# Program Finds Greatest among 2 numbers. Here the user enters two numbers and the greatest among the two numbers is found by comparing the two numbers and the result is displayed.

Here is source code of the C# Program to Find Greatest among 2 numbers. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Find Greatest among 2 numbers*
3. *\*/*
4. **using** System;
5. **class** prog
6. {
7. **public** **static** **void** Main()
8. {
9. **int** a, b;
10. Console.WriteLine("Enter the Two Numbers : ");
11. a = Convert.ToInt32(Console.ReadLine());
12. b = Convert.ToInt32(Console.ReadLine());
13. **if** (a > b)
14. {
15. Console.WriteLine("{0} is the Greatest Number", a);
16. }
17. **else**
18. {
19. Console.WriteLine("{0} is the Greatest Number ", b);
20. }
21. Console.ReadLine();
22. }
23. }

advertisements

Here is the output of the C# Program:

Enter the Two Numbers :

24

34

34 is the Greatest Number

#### 5. C# Examples on Unboxing Operation and Sealed classes.

The automatic conversion of primitive data types into its equivalent wrapper type is known as boxing and opposite operation is known as unboxing. A method, indexer, property, or event on a derived class that is overriding a virtual member of the base class can declare that member as sealed. The following C# Programs create a sealed class, performs unboxing operation, prints squarefeet of a house, lists all the prime numbers between 1 to 100 and accepts a number from the user and indicates it if it is positive.

C# Program to Display Squarefeet of a House

This C# Program Displays Squarefeet of a House. Here the length and width of the room is given and the squarefeet is calculated and displayed.

Here is source code of the C# Program to Display Squarefeet of a House. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Display Squarefeet of a House*
3. *\*/*
4. **using** System;
5. **class** pgm
6. {
7. **public** **static** **void** Main()
8. {
9. **int** length, width, area;
10. Console.Write ("Enter length of room in feet: ");
11. length = Convert.ToInt32 (Console.ReadLine());
12. Console.Write ( "Enter width of room in feet:");
13. width = Convert.ToInt32(Console.ReadLine());
14. area = length \* width;
15. Console.WriteLine ("Floor is " + area + " square feet.");
16. Console.ReadLine();
17. }
18. }

advertisements

Here is the output of the C# Program:

Enter Length of Room in Feet : 20

Enter width of Room in Feet : 20

Floor is 400 square feet.

C# Program to Create Sealed Class

This C# Program Creates Sealed Class. Here sealed classes are used to restrict the inheritance feature of object oriented programming. Once a class is defined as sealed class, this class cannot be inherited.

Here is source code of the C# Program to Create Sealed Class. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Create Sealed Class*
3. *\*/*
4. **using** System;
5. **sealed** **class** SealedClass
6. {
7. **public** **int** x;
8. **public** **int** y;
9. }
11. **class** SealedTest
12. {
13. **static** **void** Main()
14. {
15. SealedClass sc = new SealedClass();
16. sc.x = 100;
17. sc.y = 180;
18. Console.WriteLine("x = {0}, y = {1}", sc.x, sc.y);
19. Console.ReadLine();
20. }
21. }

advertisements

Here is the output of the C# Program:

x = 100 ,y = 180

C# Program to Perform Unboxing Operation

This C# Program Performs Unboxing Operation. Here Unboxing is an explicit conversion from the type object to a value type or from an interface type to a value type that implements the interface.

Here is source code of the C# Program to Perform Unboxing Operation. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Perform Unboxing Operation*
3. *\*/*
4. **using** System;
5. **class** sample
6. {
7. **int** data;
8. **void** insert(**object** x)
9. {
10. data = (**int**)x \* 5;
11. }
12. **object** delete()
13. {
14. data=0;
15. **return** (**object**)data;
16. }
17. **public** **static** **void** Main()
18. {
19. sample s = new sample();
20. s.insert(10);
21. Console.WriteLine("Data : {0}", s.data);
22. Console.WriteLine("Data : {0}", s.delete());
23. Console.ReadLine();
24. }
25. }

advertisements

Here is the output of the C# Program:

Data : 50

Data : 0

C# Program to Display All the Prime Numbers Between 1 to 100

This C# Program Displays All the Prime Numbers Between 1 to 100. Here prime number is a natural number greater than 1 that has no positive divisors other than 1 and itself.

Here is source code of the C# Program to Display All the Prime Numbers Between 1 to 100. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Display All the Prime Numbers Between 1 to 100*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
8. **namespace** PrimeNumber
9. {
10. **class** Program
11. {
12. **static** **void** Main(**string**[] args)
13. {
14. **bool** isPrime = **true**;
15. Console.WriteLine("Prime Numbers : ");
16. **for** (**int** i = 2; i <= 100; i++)
17. {
18. **for** (**int** j = 2; j <= 100; j++)
19. {
21. **if** (i != j && i % j == 0)
22. {
23. isPrime = **false**;
24. **break**;
25. }
27. }
28. **if** (isPrime)
29. {
30. Console.Write("**\t**" +i);
31. }
32. isPrime = **true**;
33. }
34. Console.ReadKey();
35. }
36. }
37. }

advertisements

Here is the output of the C# Program:

Prime Numbers :

2 3 5 7 11 13 17 19 23 29

31 37 41 43 47 53 59 61 67 71

73 79 83 89 97

C# Program to Accept a Number from the user and Display it if it is Positive

This C# Program to Accept a Number from the user and Display it if it is Positive. Here if a given number is less than zero then it is displayed as positive else negative.

Here is source code of the C# Program to Accept a Number from the user and Display it if it is Positive. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Accept a Number from the user and Display it*
3. *\* if it is Positive*
4. *\*/*
5. **using** System;
6. **class** program
7. {
8. **public** **static** **void** Main(**string**[] args)
9. {
10. Console.WriteLine("Enter a number: ");
11. **int** number = Convert.ToInt32(Console.ReadLine());
12. **if** (number > 0)
13. {
14. Console.WriteLine("Number is positive");
15. }
16. **else** **if** (number == 0)
17. {
18. Console.WriteLine("Number is 0");
19. }
20. **else**
21. {
22. Console.WriteLine("Number is negative");
23. }
24. Console.ReadLine();
25. }
26. }

advertisements

Here is the output of the C# Program:

Enter a Number : -4

Number is Negative

#### 6. C# Examples on Numerical Operations

The average of a set of numbers is the sum of the numbers divided by the number of values in the set. The following C# Programs calculates the magnitude of integer, evaluates average for the set of values and displays numbers in the form of triangle.

C# Program to Find Magnitude of Integer

This C# Program Finds Magnitude of Integer. Here Magnitude of an integer is nothing but the length of the integer which is obtained using the mod function.

Here is source code of the C# Program to Find Magnitude of Integer. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Find Magnitude of Integer*
3. *\*/*
4. **using** System;
5. **public** **class** Program
6. {
7. **public** **static** **void** Main()
8. {
9. **int** num, mag=0;
10. Console.WriteLine("Enter the Number : ");
11. num = **int**.Parse(Console.ReadLine());
12. Console.WriteLine("Number: " + num);
13. **while** (num > 0)
14. {
15. mag++;
16. num = num / 10;
17. };
18. Console.WriteLine("Magnitude: " + mag);
19. Console.Read();
20. }
21. }

advertisements

Here is the output of the C# Program:

Enter the Number : 3145678

Number : 3145678

Magnitude : 7

C# Program to Compute Average for the Set of Values

This C# Program to Compute Average for the Set of Values. Here the elements are obtained from the user and its average is found and displayed.

Here is source code of the C# Program to Compute Average for the Set of Values. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Compute Average for the Set of Values*
3. *\*/*
4. **using** System;
5. **class** program
6. {
7. **public** **static** **void** Main()
8. {
9. **int** m, i, sum = 0, avg = 0;
10. Console.WriteLine("Enter the Number of Terms in the Array ");
11. m = **int**.Parse(Console.ReadLine());
12. **int**[] a = new **int**[m];
13. Console.WriteLine("Enter the Array Elements ");
14. **for** (i = 0; i < m; i++)
15. {
16. a[i] = **int**.Parse(Console.ReadLine());
17. }
18. **for** (i = 0; i < m; i++)
19. {
20. sum += a[i];
21. }
22. avg = sum / m;
23. Console.WriteLine("Average is {0}", avg);
24. Console.ReadLine();
25. }
26. }

Here is the output of the C# Program:

Enter the Number of Terms in the Array : 4

Enter the Elements

1

2

3

4

Average is 2

C# Program to Display Numbers in the form of Triangle

This C# Program Displays Numbers in the form of Triangle. Here the numbers are displayed in the form of a triangle.

Here is source code of the C# Program to Display Numbers in the form of Triangle. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Display Numbers in the form of Triangle*
3. *\*/*
4. **using** System;
5. **class** Pascal
6. {
7. **public** **static** **void** Main()
8. {
9. **int**[,] arr = new **int**[8, 8];
10. **for** (**int** i = 0; i < 8; i++)
11. {
12. **for** (**int** k = 7; k > i; k--)
13. { *//For loop to print spaces*
14. Console.Write(" ");
15. }
17. **for** (**int** j = 0; j < i; j++)
18. {
19. **if** (j == 0 || i == j)
20. {
21. arr[i, j] = 1;
22. }
23. **else**
24. {
25. arr[i, j] = arr[i - 1, j] + arr[i - 1, j - 1];
26. }
27. Console.Write(arr[i, j] + " ");
28. }
29. Console.WriteLine();
31. }
32. Console.ReadLine();
33. }
34. }

advertisements

Here is the output of the C# Program:

1

1 1

1 2 1

1 3 3 1

1 4 6 4 1

1 5 10 10 5 1

1 6 15 20 15 6 1

#### 7. C# Examples on Interface and Looping Operations

An interface is a collection of abstract methods. The C# Program in the section implements for-each loop in the interface. The other programs implement phonebook, calculation of acceleration, finding a number using pythagoras theorem and performs division of same base exponents.

C# Program to Implement PhoneBook

This C# Program Implements PhoneBook.Reads a file with names and phone numbers and builds a phone book from it and Returns the phone number for a given name.

Here is source code to Implement PhoneBook.The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Implement PhoneBook*
3. *\*/*
4. **using** System;
5. **using** System.Collections;
6. **using** System.IO;
7. **class** PhoneBook
8. {
10. **static** **void** Main(**string**[] arg)
11. {
12. Hashtable tab = new Hashtable();
13. **string** fileName;
14. **if**
15. {
16. (arg.Length > 0) fileName = arg[0];
17. }
18. **else**
19. {
20. fileName = "phoneBook.txt";
21. }
22. StreamReader r = File.OpenText(fileName);
23. **string** line = r.ReadLine();
24. **while** (line != **null**)
25. {
26. **int** pos = line.IndexOf('=');
27. **string** name = line.Substring(0, pos).Trim();
28. **long** phone = Convert.ToInt64(line.Substring(pos + 1));
29. tab[name] = phone;
30. line = r.ReadLine();
31. }
32. r.Close();
33. **for** (; ; )
34. {
35. Console.Write("Name : ");
36. **string** name = Console.ReadLine().Trim();
37. **if** (name == "")
38. **break**;
39. **object** phone = tab[name];
40. **if** (phone == **null**)
41. Console.WriteLine("-- Not Found in Phone Book");
42. **else**
43. Console.WriteLine(phone);
44. }
45. }
46. }

advertisements

Here is the output of the C# Program:

Name : Ram

9999945670

Name : Raj

-- Not Found in Phone Book

C# Program to Implement for-each in Inteface

This C# Program Implements for-each in Inteface. It demonstrates the best practice for iterating a custom collection by implementing the IEnumerable and IEnumerator interfaces. In this code,members of these interfaces are not explicitly called, but they are implemented to support the use of foreach to iterate through the collection.

Here is source code of the C# Program to Implement for-each in Inteface. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Implement for-each in Inteface*
3. *\*/*
4. **using** System;
5. **using** System.Collections;
6. **class** GrowableArray : IEnumerable
7. {
8. **object**[] a;
9. **public** GrowableArray(**int** size)
10. {
11. a = new **object**[size];
12. }
13. **public** GrowableArray() : **this**(8) {}
14. **void** Grow()
15. {
16. **object**[] b = new **object**[2 \* a.Length];
17. Array.Copy(a, b, a.Length);
18. a = b;
19. }
20. **public** **object** **this**[**int** i]
21. {
22. **set**
23. {
24. **if** (i >= a.Length) Grow();
25. a[i] = **value**;
26. }
27. **get**
28. {
29. **if** (i >= a.Length) Grow();
30. **return** a[i];
31. }
32. }
33. **public** IEnumerator GetEnumerator()
34. {
35. **return** new GAEnumerator(a);
36. }
37. **class** GAEnumerator : IEnumerator
38. {
39. **object**[] a;
40. **int** i = -1;
41. **public** GAEnumerator(**object**[] a) { **this**.a = a; }
42. **public** **object** Current
43. {
44. **get**
45. {
46. **return** a[i];
47. }
48. }
49. **public** **void** Reset()
50. {
51. i = -1;
52. }
53. **public** **bool** MoveNext()
54. {
55. **do** i++;
56. **while** (i < a.Length && a[i] == **null**);
57. **if** (i == a.Length)
58. **return** **false**;
59. **else** **return** **true**;
60. }
61. }
63. }
64. **class** Test
65. {
66. **public** **static** **void** Main()
67. {
68. GrowableArray a = new GrowableArray(2);
69. a[0] = 0;
70. a[1] = 1;
71. a[3] = 3;
72. **foreach** (**object** x **in** a) Console.Write(" " + x);
73. }

advertisements

Here is the output of the C# Program:

Demonstrating foreach Interface by Displaying Numbers from 100 to 105 :

100

101

102

103

104

105

C# Program to Calculate Acceleration

This C# Program Calculates Acceleration. Here the velocity and time are obtained and acceleration is calculated.

Here is source code of the C# Program to Calculate Acceleration. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Calculate Acceleration*
3. *using System;*
4. *class program*
5. *{*
6. *static void Main(string[] args)*
7. *{*
8. *int v, t, acc;*
9. *Console.WriteLine("Enter the Velocity : ");*
10. *v = int.Parse(Console.ReadLine());*
11. *Console.WriteLine("Enter the Time : ");*
12. *t = int.Parse(Console.ReadLine());*
13. *acc = v / t;*
14. *Console.WriteLine("Acceleration : {0}", acc);*
15. *}*
16. *}*

advertisements

Here is the output of the C# Program:

Enter the Velocity :

10

Enter the Time :

2

Acceleration : 5

C# Program to Find a Number using Pythagoras Theorem

This C# Program Finds a Number using Pythagoras Theorem. Here the pythagoras theorem is used to find the other number.

Here is source code of theC# Program to Find a Number using Pythagoras Theorem. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Find a Number using Pythagoras Theorem*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
8. **class** Program
9. {
10. **static** **void** Main(**string**[] args)
11. {
12. **double** a, b, c;
13. Console.WriteLine("Enter the First Value ");
14. a = **double**.Parse(Console.ReadLine());
15. Console.WriteLine("Enter the Second Value ");
16. b = **double**.Parse(Console.ReadLine());
17. c = Math.Sqrt(a \* a + b \* b);
18. Console.WriteLine("The Other Number is : {0}", c);
19. Console.ReadLine();
21. }
22. }

advertisements

Here is the output of the C# Program:

Enter the First Value

3

Enter the Second Value

4

The Other Number is : 5

C# Program to Perform Division of Exponents of Same Base

This C# Program Performs Division of Exponents of Same Base. Here the power value for all same base is found.

Here is source code of the C# Program to Perform Multiplication of Exponents of Same Base. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Perform Division of Exponents of Same Base*
3. *\*/*
4. **using** System;
5. **class** Program
6. {
7. **static** **void** Main()
8. {
9. Console.WriteLine("Enter the Base : ");
10. **double** num = **double**.Parse(Console.ReadLine());
11. Console.WriteLine("Enter the First Exponent :");
12. **double** exp1 = **double**.Parse(Console.ReadLine());
13. Console.WriteLine("Enter the Second Exponent :");
14. **double** exp2 = **double**.Parse(Console.ReadLine());
15. **double** div;
16. div = exp1 - exp2;
17. Console.WriteLine("Result is : {0}^{1} : {2}", num, div, Math.Pow(num, div));
18. Console.ReadLine();
19. }
20. }

advertisements

Here is the output of the C# Program:

Enter the Base :

2

Enter the First Exponent :

4

Enter the Second Exponent :

3

Result is : 2^1 : 2

#### B. C# Programming Examples on Inheritance

The following section contains C# programs on Inheritance and various illustrations on inheritance.

#### C# Examples Demonstrating the Different Types of Inheritance

When a class is derived from another derived class it is called Multilevel Inheritance. In Multilevel Inheritance, there is one base class and the remaining two are derived classes. In Single Inheritance, subclasses inherit the features of a single super class. Hierarchical Inheritance is a method of inheritance where one or more derived classes is derived from common base class. The following programs illustrate Multilevel Inheritance, Single Inheritance, Hierarchical Inheritance and displays the cost of a rectangle plot using Inheritance.

C# Program to Demonstrate Multilevel Inheritance

This C# Program Demonstrates Multilevel Inheritance. Here when a derived class is created from another derived class, then that inheritance is called as multi level inheritance.

Here is source code of the C# Program to Demonstrate Multilevel Inheritance. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Demonstrate Multilevel Inheritance*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
8. **namespace** Inherit
9. {
10. **class** inheri : vehicle
11. {
12. **public** **void** Noise()
13. {
14. Console.WriteLine("All Vehicles Creates Noise !! ");
15. }
16. **static** **void** Main(**string**[] args)
17. {
18. inheri obj = new inheri();
19. obj.mode();
20. obj.feature();
21. obj.Noise();
22. Console.Read();
23. }
24. }
25. **class** Mode
26. {
27. **public** **void** mode()
28. {
29. Console.WriteLine("There are Many Modes of Transport !!");
30. }
31. }
32. **class** vehicle : Mode
33. {
34. **public** **void** feature()
35. {
36. Console.WriteLine("They Mainly Help in Travelling !!");
37. }
38. }
39. }

Here is the output of the C# Program:

There are Many Modes of Transport !!

They Mainly Help in Travelling !!

All Vehicles Creates Noise !!

C# Program to Illustrate Single Inheritance

This C# Program Illustrates Single Inheritance. Here In single inheritance we have single base class that is inherited by the derived class and the derived class has all the features of the base class and can add some new features and italso depends on the access specifier that is used at the time of base class inheritance.

Here is source code of the C# Program to Caluculate the power exponent value. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Illustrate Single Inheritance*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
8. **namespace** Inheritance
9. {
10. **class** Program
11. {
12. **static** **void** Main(**string**[] args)
13. {
14. Teacher d = new Teacher();
15. d.Teach();
16. Student s = new Student();
17. s.Learn();
18. s.Teach();
19. Console.ReadKey();
20. }
21. **class** Teacher
22. {
23. **public** **void** Teach()
24. {
25. Console.WriteLine("Teach");
26. }
27. }
28. **class** Student : Teacher
29. {
30. **public** **void** Learn()
31. {
32. Console.WriteLine("Learn");
33. }
34. }
35. }
36. }

advertisements

Here is the output of the C# Program:

Teach

Learn

Teach

C# Program to Illustrate Hierarchical Inheritance

This C# Program Illustrates Hierarchical Inheritance. Here Single parent and multiple child and when more than one derived class are created from single base class is called C# Hierarchical Inheritance.

Here is source code of the C# Program to Illustrate Hierarchical Inheritance. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Illustrate Hierarchical Inheritance*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
8. **namespace** Inheritance
9. {
10. **class** Program
11. {
12. **static** **void** Main(**string**[] args)
13. {
14. Principal g = new Principal();
15. g.Monitor();
16. Teacher d = new Teacher();
17. d.Monitor();
18. d.Teach();
19. Student s = new Student();
20. s.Monitor();
21. s.Learn();
22. Console.ReadKey();
23. }
24. **class** Principal
25. {
26. **public** **void** Monitor()
27. {
28. Console.WriteLine("Monitor");
29. }
30. }
31. **class** Teacher : Principal
32. {
33. **public** **void** Teach()
34. {
35. Console.WriteLine("Teach");
36. }
37. }
38. **class** Student : Principal
39. {
40. **public** **void** Learn()
41. {
42. Console.WriteLine("Learn");
43. }
44. }
45. }
46. }

advertisements

Here is the output of the C# Program:

Monitor

Monitor

Teach

Monitor

Learn

C# Program to Illustrate Multilevel Inheritance with Virtual Methods

This C# Program Illustrates Multilevel Inheritance with Virtual Methods. Here the system executes the first override-virtual method found in the hierarchy.

Here is source code of the C# Program to Illustrate Multilevel Inheritance with Virtual Methods. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Illustrate Multilevel Inheritance with Virtual Methods*
3. *\*/*
4. **using** System;
5. **public** **class** Person
6. {
7. **protected** **string** RNO = "44";
8. **protected** **string** name = "Ram";
9. **public** **virtual** **void** GetInfo()
10. {
11. Console.WriteLine("Name: {0}", name);
12. Console.WriteLine("RNO: {0}", RNO);
13. Console.WriteLine();
14. }
15. }
16. **class** Student : Person
17. {
18. **public** **string** id = "ABC";
19. **public** **override** **void** GetInfo()
20. {
21. **base**.GetInfo();
22. Console.WriteLine("Student ID: {0}", id);
23. }
24. }
25. **class** Stud : Student
26. {
27. **private** **string** StudentAddress = "USA";
28. **public** **void** GetInfo()
29. {
30. **base**.GetInfo();
31. Console.WriteLine("Student Address: {0}", StudentAddress);
32. }
33. }
34. **class** TestClass
35. {
36. **public** **static** **void** Main()
37. {
38. Student E = new Student();
39. E.GetInfo();
40. Stud Stud = new Stud();
41. Stud.GetInfo();
42. Console.ReadLine();
43. }
44. }

advertisements

Here is the output of the C# Program:

Name : Ram

RNO : 44

Student ID : ABC

Name : Ram

RNO : 44

Student ID : ABC

Student Address : USA

C# Program to Display Cost of a Rectangle Plot Using Inheritance

This C# Program Displays Cost of a Rectangle Plot Using Inheritance. Here cost of the rectangle is found using inheritance

Here is source code of the C# Program to Display Cost of a Rectangle Plot Using Inheritance. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Display Cost of a Rectangle Plot Using Inheritance*
3. *\*/*
4. **using** System;
5. **class** Rectangle
6. {
7. **protected** **double** length;
8. **protected** **double** width;
9. **public** Rectangle(**double** l, **double** w)
10. {
11. length = l;
12. width = w;
13. }
14. **public** **double** GetArea()
15. {
16. **return** length \* width;
17. }
18. **public** **void** Display()
19. {
20. Console.WriteLine("Length: {0}", length);
21. Console.WriteLine("Width: {0}", width);
22. Console.WriteLine("Area: {0}", GetArea());
23. }
24. }
25. **class** Tabletop : Rectangle
26. {
27. **private** **double** cost;
28. **public** Tabletop(**double** l, **double** w)
29. : **base**(l, w)
30. { }
31. **public** **double** costcal()
32. {
33. **double** cost;
34. cost = GetArea() \* 70;
35. **return** cost;
36. }
37. **public** **void** Display()
38. {
39. **base**.Display();
40. Console.WriteLine("Cost: {0}", costcal());
41. }
42. }
43. **class** CalRectangle
44. {
45. **static** **void** Main(**string**[] args)
46. {
47. Tabletop t = new Tabletop(7.5, 8.03);
48. t.Display();
49. Console.ReadLine();
50. }
51. }

advertisements

Here is the output of the C# Program:

Length: 7.5

Width: 8.03

Area: 60.225

Cost: 4215.75

#### C. C# Programming Examples on LINQ

This section contains C# programs on the applications of LINQ and usage of LINQ in SQL clauses.

#### 1. C# Examples on Applications of LINQ

LINQ is language integrated query which allow programmers to query various objects without worrying about underlying implementations of these objects. File extension is the suffix that is given to a filename. Numbers ending in 1,3,5,7,9 are called Odd numbers. IEnumerable is the Interface that implements GetEnumerator. The method GetEnumerator is called when the class implements IEnumerable and a new object is returned which can be used in the foreach loop. The following programs perform counting anf grouping of file extensions, determines the file size, performs implementation of IEnumerable Interface and gives a set of odd numbers using LINQ.

C# Program to Count File Extensions and Group it using LINQ

This C# Program Counts File Extensions and Group it using LINQ. Here a service reads files generated in a folder every hour and returns a string array containing the file names and showes the count of files grouped by the file extension.

Here is source code of the C# Program to Count File Extensions and Group it using LINQ. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Count File Extensions and Group it using LINQ*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
8. **using** System.IO;
9. **namespace** ConsoleApplication9
10. {
11. **class** Program
12. {
13. **public** **static** **void** Main()
14. {
15. **string**[] arr = { "aaa.txt", "bbb.TXT", "xyz.abc.pdf", "aaaa.PDF", "abc.xml", "ccc.txt", "zzz.txt" };
16. **var** egrp = arr.**Select**(file => Path.GetExtension(file).TrimStart('.').ToLower())
17. .GroupBy(x => x,(ext, extCnt) =>new
18. {
19. Extension = ext,
20. Count = extCnt.Count()
21. });
23. **foreach** (**var** v **in** egrp)
24. Console.WriteLine("{0} File(s) with {1} Extension ",v.Count, v.Extension);
25. Console.ReadLine();
26. }
27. }
28. }

advertisements

Here is the output of the C# Program:

4 File(s) with txt Extension

2 File(s) with pdf Extension

1 File(s) with xml Extension

C# Program to Calculate Size of File using LINQ

This C# Program Calculates Size of File using LINQ. Here the size of the folder is found using LINQ functions.

Here is source code of the C# Program to Calculate Size of File using LINQ. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Calculate Size of File using LINQ*
3. *\*/*
4. **using** System;
5. **using** System.Linq;
6. **using** System.IO;
7. **class** Program
8. {
9. **static** **void** Main(**string**[] args)
10. {
11. **string**[] dirfiles = Directory.GetFiles("c:**\\**sri**\\**");
12. **var** avg = dirfiles.**Select**(file =>new FileInfo(file).Length).Average();
13. avg = Math.Round(avg / 10, 1);
14. Console.WriteLine("The Average file size is {0} MB",avg);
15. Console.ReadLine();
16. }
17. }

advertisements

Here is the output of the C# Program:

The Average file size is 8.8 MB

C# Program to Generate Odd Numbers in Parallel using LINQ

This C# Program Generates Odd Numbers in Parallel using LINQ. Here it Provides a set of methods for querying objects that implement ParallelQuery{TSource} to generate odd numbers in parallel.

Here is source code of the C# Program to Generate Odd Numbers in Parallel using LINQ. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Generate Odd Numbers in Parallel using LINQ*
3. *\*/*
4. **using** System;
5. **using** System.Linq;
6. **using** System.Collections.Generic;
8. **class** Program
9. {
10. **static** **void** Main(**string**[] args)
11. {
12. IEnumerable<**int**> oddNums = ((ParallelQuery<**int**>)ParallelEnumerable.Range(20, 2000))
13. .**Where**(x => x % 2 != 0)
14. .**Select**(i => i);
15. **foreach** (**int** n **in** oddNums) { Console.WriteLine(n); }
16. Console.ReadLine();
17. }
18. }

advertisements

Here is the output of the C# Program:

871

1371

1871

373

873

1373

1873

375

875

1375

1875

377

877

1377

1877

379

879

1379

1879

381

881

1381

1881

383

883

1383

1883

385

885

1385

1885

387

887

1387

1887

389

889

1389

1889

391

891

1391

1891

393

893

1393

1893

395

895

1395

1895

397

897

1397

1897

399

899

1399

1899

401

901

1401

1901

403

903

1403

1903

405

905

1405

1905

407

907

1407

1907

409

909

1409

1909

411

911

1411

1911

413

913

1413

1913

415

915

1415

1915

417

917

1417

1917

419

919

1419

1919

421

921

1421

1921

423

923

1423

1923

425

925

1425

1925

427

927

1427

1927

429

929

1429

1929

431

931

1431

1931

433

933

1433

1933

435

935

1435

1935

437

937

1437

1937

439

939

1439

1939

441

941

1441

1941

443

943

1443

1943

445

945

1445

1945

447

947

1447

1947

449

949

1449

1949

451

951

1451

1951

453

953

1453

1953

455

955

1455

1955

457

957

1457

1957

459

959

1459

1959

461

961

1461

1961

463

963

1463

1963

465

965

1465

1965

467

967

1467

1967

469

969

1469

1969

471

971

1471

1971

473

973

1473

1973

475

975

1475

1975

477

977

1477

1977

479

979

1479

1979

481

981

1481

1981

483

983

1483

1983

485

985

1485

1985

487

987

1487

1987

489

989

1489

1989

491

991

1491

1991

493

993

1493

1993

495

995

1495

1995

497

997

1497

1997

499

999

1499

1999

501

1001

1501

2001

503

1003

1503

2003

505

1005

1505

2005

507

1007

1507

2007

509

1009

1509

2009

511

1011

1511

2011

513

1013

1513

2013

515

1015

1515

2015

517

1017

1517

2017

519

1019

1519

2019

C# Program to Implement IEnumerable Interface using LINQ

This C# Program Implements IEnumerable Interface using LINQ. Here it exposes an enumerator, which supports a simple iteration over a non-generic collection.

Here is source code of the C# Program to Implement IEnumerable Interface using LINQ. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Implement IEnumerable Interface using LINQ*
3. *\*/*
4. **using** System;
5. **using** System.IO;
6. **using** System.Collections;
7. **using** System.Linq;
8. **class** program
9. {
10. **public** **static** **void** Main(**string**[] args)
11. {
12. **var** t = typeof(IEnumerable);
13. **var** typesIEnum = AppDomain.CurrentDomain.GetAssemblies().SelectMany(x => x.GetTypes()).**Where**(x => t.IsAssignableFrom(x));
14. **foreach** (**var** types **in** typesIEnum)
15. {
16. Console.WriteLine(types.FullName);
17. }
18. Console.ReadLine();
19. }
20. }

advertisements

Here is the output of the C# Program:

System.Linq.Parallel.IndexedSelectQueryOperator`2

System.Linq.Parallel.IndexedSelectQueryOperator`2+IndexedSelectQueryOperatorResults

System.Linq.Parallel.IndexedWhereQueryOperator`1

System.Linq.Parallel.LastQueryOperator`1

System.Linq.Parallel.ReverseQueryOperator`1

System.Linq.Parallel.ReverseQueryOperator`1+ReverseQueryOperatorResults

System.Linq.Parallel.SelectManyQueryOperator`3

System.Linq.Parallel.SelectQueryOperator`2

System.Linq.Parallel.SelectQueryOperator`2+SelectQueryOperatorResults

System.Linq.Parallel.SingleQueryOperator`1

System.Linq.Parallel.SortQueryOperator`2

System.Linq.Parallel.SortQueryOperatorResults`2

System.Linq.Parallel.TakeOrSkipQueryOperator`1

System.Linq.Parallel.TakeOrSkipQueryOperator`1+TakeOrSkipQueryOperatorResults

System.Linq.Parallel.TakeOrSkipWhileQueryOperator`1

System.Linq.Parallel.WhereQueryOperator`1

System.Linq.Parallel.ListChunk`1

System.Linq.Parallel.Lookup`2

C# Program to Divide Sequence into Groups using LINQ

This C# Program Divides Sequence into Groups using LINQ. Here it first projects each element of a sequence into a new form and group it.The results are shaped into an enumerable collection of anonymous objects with a property Min and Max.

Here is source code of the C# Program to Divide Sequence into Groups using LINQ. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Divide Sequence into Groups using LINQ*
3. *\*/*
4. **using** System;
5. **using** System.Linq;
6. **using** System.IO;
7. **class** Program
8. {
9. **static** **void** Main(**string**[] args)
10. {
11. **var** seq = Enumerable.Range(100, 100).**Select**(x => x / 10f);
12. **var** grps = **from** x **in** seq.**Select**((i, j) => new { i, Grp = j / 10 })
13. **group** x.i **by** x.Grp **into** y
14. **select** new { Min = y.Min(), Max = y.Max() };
15. **foreach** (**var** grp **in** grps)
16. Console.WriteLine("Min: " + grp.Min + " Max:" + grp.Max);
17. Console.ReadLine();
18. }
19. }

advertisements

Here is the output of the C# Program:

Min : 10 Max : 10.9

Min : 11 Max : 11.9

Min : 12 Max : 12.9

Min : 13 Max : 13.9

Min : 14 Max : 14.9

Min : 15 Max : 15.9

Min : 16 Max : 16.9

Min : 17 Max : 17.9

Min : 18 Max : 18.9

Min : 19 Max : 19.9

#### 2. C# Examples on the Usage of LINQ in SQL Clauses

The FROM clause provides a set of tables that contain information that the queries refer to for selection. WHERE clause specifies the conditions on the basis of which rows are returned. SELECT clause provides the fields and expressions to be displayed into the results of a query. The following programs show the usage of SELECT, FROM and WHERE clauses by printing the student details in a class, printing the largest and smallest numbers in an array.

C# Program to Display the Student Details using Select Clause LINQ

This C# Program Displays the Student Details using Select Clause LINQ. Here select clause specifies the type of values that will be produced when the query is executed.

Here is source code of the C# Program to Display the Student Details using Select Clause LINQ. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Display the Student Details using Select Clause LINQ*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
8. **using** System.Threading.Tasks;
9. **class** Programs
10. {
11. **public** **class** Student
12. {
13. **public** **string** First { **get**; **set**; }
14. **public** **string** Last { **get**; **set**; }
15. **public** **int** ID { **get**; **set**; }
16. **public** List<**int**> Marks;
17. **public** ContactInfo GetContactInfo(Programs pg, **int** id)
18. {
19. ContactInfo allinfo =
20. (**from** ci **in** pg.contactList
21. **where** ci.ID == id
22. **select** ci)
23. .FirstOrDefault();
25. **return** allinfo;
26. }
28. **public** **override** **string** ToString()
29. {
30. **return** First + "" + Last + " : " + ID;
31. }
32. }
34. **public** **class** ContactInfo
35. {
36. **public** **int** ID { **get**; **set**; }
37. **public** **string** Email { **get**; **set**; }
38. **public** **string** Phone { **get**; **set**; }
39. **public** **override** **string** ToString() { **return** Email + "," + Phone; }
40. }
42. **public** **class** ScoreInfo
43. {
44. **public** **double** Average { **get**; **set**; }
45. **public** **int** ID { **get**; **set**; }
46. }
47. List<Student> students = new List<Student>()
48. {
49. new Student {First="Tom", Last=".S", ID=1, Marks= new List<**int**>() {97, 92, 81, 60}},
50. new Student {First="Jerry", Last=".M", ID=2, Marks= new List<**int**>() {75, 84, 91, 39}},
51. new Student {First="Bob", Last=".P", ID=3, Marks= new List<**int**>() {88, 94, 65, 91}},
52. new Student {First="Mark", Last=".G", ID=4, Marks= new List<**int**>() {97, 89, 85, 82}},
53. };
54. List<ContactInfo> contactList = new List<ContactInfo>()
55. {
56. new ContactInfo {ID=111, Email="Tom@abc.com", Phone="9328298765"},
57. new ContactInfo {ID=112, Email="Jerry123@aaa.com", Phone="9876543201"},
58. new ContactInfo {ID=113, Email="Bobstar@aaa.com", Phone="9087467653"},
59. new ContactInfo {ID=114, Email="Markantony@qqq.com", Phone="9870098761"}
60. };

63. **static** **void** Main(**string**[] args)
64. {
65. Programs pg = new Programs();
67. IEnumerable<Student> studentQuery1 =
68. **from** student **in** pg.students
69. **where** student.ID > 1
70. **select** student;
72. Console.WriteLine("Query : Select range\_variable");
73. Console.WriteLine("Name : ID");
74. **foreach** (Student s **in** studentQuery1)
75. {
76. Console.WriteLine(s.ToString());
77. }
78. Console.ReadLine();
79. }
80. }

advertisements

Here is the output of the C# Program:

Enter the Number : 2

Enter the Exponent :3

Result : 8

C# Program to Display the Greatest numbers in an Array using WHERE Clause LINQ

This C# Program Displays the Greatest numbers in an Array using WHERE Clause LINQ. Here the where clause is used in a query expression to specify which elements from the data source will be returned in the query expression

Here is source code of the C# Program to Display the Greatest numbers in an Array using WHERE Clause LINQ. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Display the Greatest numbers in an Array using WHERE Clause LINQ*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
8. **using** System.Threading.Tasks;
10. **class** Program
11. {
12. **static** **void** Main()
13. {
14. **int**[] numbers = { 500, 344, 221, 4443, 229, 1008, 6000, 767, 256, 0 };
15. **var** greaterNums =
16. **from** num **in** numbers
17. **where** num > 500
18. **select** num;
19. Console.WriteLine("Numbers Greater than 500 :");
20. **foreach** (**var** s **in** greaterNums)
21. {
22. Console.Write(s.ToString() + " ");
23. }
24. Console.Read();
25. }
26. }

advertisements

Here is the output of the C# Program:

Numbers Greater than 500 :

4443 1008 6000 767

C# Program to Display the Smallest numbers in an Array using FROM Clause LINQ

This C# Program Displays the Smallest numbers in an Array using FROM Clause LINQ. Here the from clause allows to obtain the result of an expression inside the query expression.

Here is source code of the C# Program to Display the Smallest numbers in an Array using FROM Clause LINQ. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Display the Smallest numbers in an Array using FROM Clause LINQ*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
8. **using** System.Threading.Tasks;
10. **namespace** ConsoleApplication2
11. {
12. **class** program
13. {
14. **static** **void** Main()
15. {
16. **int**[] numbers = { 50,30,45,10,60,100,500,300,40,22,44,55,66,1000 };
17. **var** program = **from** num **in** numbers
18. **where** num < 50
19. **select** num;
20. Console.WriteLine("Numbers less than 50 are :");
21. **foreach** (**int** i **in** program)
22. {
23. Console.Write(i + " ");
24. }
25. Console.ReadLine();
26. }
27. }
28. }

advertisements

Here is the output of the C# Program:

Numbers less than 50 are :

30 45 10 40 22 44

C# Program to Implement Let Condition using LINQ

This C# Program Implements Let Condition using LINQ. Here the Let clause allows to store the result of an expression inside the query expression.

Here is source code of the C# Program to Implement Let Condition using LINQ. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Implement Let Condition using LINQ*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
9. **class** Student
10. {
11. **public** **string** Name { **get**; **set**; }
12. **public** **string** Regno { **get**; **set**; }
13. **public** **int** Marks { **get**; **set**; }
15. }
16. **class** Program
17. {
18. **static** **void** Main(**string**[] args)
19. {
20. *//Object Initialization for Student class*
21. List<Student> objStudent = new List<Student>{
22. new Student{ Name="Tom",Regno="R001",Marks=80},
23. new Student{ Name="Bob",Regno="R002",Marks=40},
24. new Student{ Name="jerry",Regno="R003",Marks=25},
25. new Student{ Name="Syed",Regno="R004",Marks=30},
26. new Student{ Name="Mob",Regno="R005",Marks=70},
27. };
29. **var** objresult = **from** stu **in** objStudent
30. let totalMarks = objStudent.Sum(mark => mark.Marks)
31. let avgMarks = totalMarks / 5
32. **where** avgMarks > stu.Marks
33. **select** stu;
34. **foreach** (**var** stu **in** objresult)
35. {
36. Console.WriteLine("Student: {0} {1}", stu.Name, stu.Regno);
38. }
39. Console.ReadLine();
40. }
41. }

advertisements

Here is the output of the C# Program:

Student: Bob R002

Student: jerry R003

Student: Syed R004

#### D. C# Programming Examples on Arrays

This section contains C# programs on types of arrays, indexing, sort, search, copy reverse operations and negation and average operations.

#### 1. C# Examples on different types of Arrays

An array is a variable that holds fixed number of values of a certain type. The no of elements an array can hold is called the length or the size of an array. It is used for looping and accessing over the last element. One dimensional array is the simplest type of array in C# that contains only one row to store value in same data type. A two-dimensional array contains two indices to select an element. An array whose elements are again arrays is called a Jagged Array. The elements of a jagged array are of different dimensions and sizes. The number of dimensions of an array is called its rank. Each dimension in an array has an upper and lower bound, which gives the range of values that can be used as subscripts for that dimension. The programs in this section perform conversion of a 2D array into 1D array, finds the length array length, determines the upper bound and lower bound of an array, evaluates the rank of a given array and demonstrates the functionalities of a jagged array.

C# Program to Convert a 2D Array into 1D Array

This C# Program Converts a 2D Array into 1D Array. Here the elements of the 2-Dimensional matrix are obtained from the user and are then converted and displayed as a 1-Dimensional array.

Here is source code of the C# Program to Convert a 2D Array into 1D Array. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Convert a 2D Array into 1D Array*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
9. **namespace** Program
10. {
11. **class** twodmatrix
12. {
13. **int** m, n;
14. **int**[,] a;
15. **int**[] b;
16. twodmatrix(**int** x, **int** y)
17. {
18. m = x;
19. n = y;
20. a = new **int**[m, n];
21. b = new **int**[m \* n];
22. }
23. **public** **void** readmatrix()
24. {
25. **for** (**int** i = 0; i < m; i++)
26. {
27. **for** (**int** j = 0; j < n; j++)
28. {
29. Console.WriteLine("a[{0},{1}]=", i, j);
30. a[i, j] = Convert.ToInt32(Console.ReadLine());
31. }
32. }
33. }
34. **public** **void** printd()
35. {
36. **for** (**int** i = 0; i < m; i++)
37. {
38. **for** (**int** j = 0; j < n; j++)
39. {
40. Console.Write("{0}**\t**", a[i, j]);
42. }
43. Console.Write("**\n**");
44. }
45. }
46. **public** **void** convert()
47. {
48. **int** k = 0;
49. **for** (**int** i = 0; i < m; i++)
50. {
51. **for** (**int** j = 0; j < n; j++)
52. {
53. b[k++] = a[i, j];
54. }
55. }
56. }
57. **public** **void** printoned()
58. {
59. **for** (**int** i = 0; i < m \* n; i++)
60. {
61. Console.WriteLine("{0}**\t**", b[i]);
62. }
63. }

66. **public** **static** **void** Main(**string**[] args)
67. {
68. twodmatrix obj = new twodmatrix(2,3);
69. Console.WriteLine("Enter the Elements : ");
70. obj.readmatrix();
71. Console.WriteLine("**\t\t** Given 2-D Array(Matrix) is : ");
72. obj.printd();
73. obj.convert();
74. Console.WriteLine("**\t\t** Converted 1-D Array is : ");
75. obj.printoned();
76. Console.ReadLine();
77. }
78. }
79. }

advertisements

Here is the output of the C# Program:

Enter the Elements :

a[0,0]=3

a[0,1]=7

a[0,2]=1

a[1,0]=9

a[1,1]=34

a[1,2]=23

Given 2-D Array(Matrix) is :

1 4 3

7 3 8

Converted 1-D Array is :

1

4

3

7

3

8

C# Program to get the Length of the Array

This C# Program gets the Length of the Array. Here the length of 1-Dimensional and 2-Dimensional array is found using the Length method.

Here is source code of the C# Program to get the Length of the Array. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to get the Length of the Array*
3. *\*/*
4. **using** System;
5. **class** Program
6. {
7. **static** **void** Main()
8. {
10. **int**[] arrayA = new **int**[5];
11. **int** lengthA = arrayA.Length;
12. Console.WriteLine("Length of ArrayA : {0}", +lengthA);
13. **long** longLength = arrayA.LongLength;
14. Console.WriteLine("Length of the LongLength Array : {0}",longLength);
15. **int**[,] twoD = new **int**[5, 10];
16. Console.WriteLine("The Size of 2D Array is : {0}",twoD.Length);
17. Console.ReadLine();
18. }
19. }

advertisements

Here is the output of the C# Program:

Length of ArrayA : 5

Length of the LongLength Array is :5

The Size of 2D Array is : 50

C# Program to Get Lower Bound and Upper Bound of an Array

This C# Program Gets Lower Bound and Upper Bound of an Array. Here The GetLowerBound and GetUppperBound methods return the lower and upper bounds of an array respectively.

Here is source code of the C# Program to Get Lower Bound and Upper Bound of an Array. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program Gets Lower Bound and Upper Bound of an Array*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
8. **namespace** lower
9. {
10. **class** Program
11. {
12. **static** **void** Main(**string**[] args)
13. {
14. Array stringArray = Array.CreateInstance(typeof(**String**), 6);
15. stringArray.SetValue("Mango", 0);
16. stringArray.SetValue("Orange", 1);
17. stringArray.SetValue("Apple", 2);
18. stringArray.SetValue("Grape", 3);
19. stringArray.SetValue("Cherry", 4);
20. stringArray.SetValue("WaterMelon", 4);
21. Console.WriteLine("The Lower Bound of the Array : {0}",stringArray.GetLowerBound(0).ToString());
22. Console.WriteLine("The Upper Bound of the Array : {0}",stringArray.GetUpperBound(0).ToString());
23. Console.ReadLine();
25. }
26. }
27. }

advertisements

Here is the output of the C# Program:

The Lower Bound of the Array : 0

The Upper Bound of the Array : 5

C# Program to Find the Rank of a given Array

This C# Program Finds the Rank of a given Array. Here the rank denotes the dimensions of the array.Thus the rank is found for various types of array that are declared.

Here is source code of the C# Program to Find the Rank of a given Array. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Find the Rank of a given Array*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
8. **using** System.Threading.Tasks;
9. **namespace** CSharpDemoApps
10. {
11. **class** Program
12. {
13. **static** **void** Main(**string**[] args)
14. {
15. **int**[] array1 = new **int**[5] { 1, 2, 3, 4, 5 };
16. **int**[,] array2 = new **int**[10, 3];
17. Console.WriteLine("Total dimensions of Array is : " + array1.Rank);
18. Console.WriteLine("Total dimensions of Array is : " + array2.Rank);
19. Console.ReadLine();
20. }
21. }
22. }

advertisements

Here is the output of the C# Program:

Total dimensions of Array is : 1

Total dimensions of Array is : 2

C# Program to Demonstrate Jagged Arrays

This C# Program Demonstrates Jagged Arrays. Here Jagged Arrays can store efficiently many rows of varying lengths. Any type of data, reference or value, can be used.

Here is source code of the C# Program to Demonstrate Jagged Arrays. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Demonstrate Jagged Arrays*
3. *\*/*
4. **using** System;
5. **class** Program
6. {
7. **static** **void** Main()
8. {
9. **int**[][] jag = new **int**[3][];
10. jag[0] = new **int**[2];
11. jag[0][0] = 11;
12. jag[0][1] = 12;
13. jag[1] = new **int**[1] {11};
14. jag[2] = new **int**[3] { 14,15, 16 };
15. **for** (**int** i = 0; i < jag.Length; i++)
16. {
17. **int**[] innerArray = jag[i];
18. **for** (**int** a = 0; a < innerArray.Length; a++)
19. {
20. Console.WriteLine(innerArray[a] + " ");
21. }
22. }
23. Console.Read();
24. }
25. }

advertisements

Here is the output of the C# Program:

11

12

11

14

15

16

#### 2. C# Examples on Indexing

Indexers treat the objects same like arrays. Array indexing uses of square brackets ([]) to index array values. The programs in the following section takes two input arrays and produces a third array by joining one array to another. The remaining programs in the section searches for an array element using array indices, demonstrates the usage of array indexers and finds the maximum and minimum numbers(largest and smallest) in the given array.

C# Program to Get 2 Arrays as Input and Produce a 3rd Array by Appending one to other

This C# Program Gets 2 Arrays as Input and Produce a 3rd Array by Appending one to other. Here Buffer.BlockCopy is used to merge two int arrays. This method acts upon bytes, not elements .

Here is source code of the C# Program to Get 2 Arrays as Input and Produce a 3rd Array by Appending one to other. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Get 2 Arrays as Input and Produce a 3rd Array by Appending one to*
3. *\* other*
4. *\*/*
5. **using** System;
7. **class** Program
8. {
9. **static** **void** Main()
10. {
11. **int**[] array1 = new **int**[5];
12. **int**[] array2 = new **int**[5];
13. **int**[] array3 = new **int**[array1.Length + array2.Length];
14. Console.WriteLine("Enter Any 5 Elements for the First Array :");
15. **for** (**int** i = 0; i < 5; i++)
16. {
17. array1[i] = **int**.Parse(Console.ReadLine());
18. }
19. Console.WriteLine("Enter Any 5 Elements for the Second Array :");
20. **for** (**int** i = 0; i < 5; i++)
21. {
22. array2[i] = **int**.Parse(Console.ReadLine());
23. }
24. Buffer.BlockCopy(array1,0,array3,0,array1.Length \* sizeof(**int**));
25. Buffer.BlockCopy(array2,0,array3,array1.Length \* sizeof(**int**),array2.Length \* sizeof(**int**));
26. Console.WriteLine("Elements in the Third Array After Appending First and Second Arrays :");
27. **foreach** (**int** **value** **in** array3)
28. {
29. Console.WriteLine(**value**);
30. }
31. Console.ReadLine();
32. }
33. }

advertisements

Here is the output of the C# Program:

Enter Any 5 Elements for the First Array :

1

2

3

4

5

Enter Any 5 Elements for the Second Array :

6

7

8

9

10

Elements in the Third Array After Appending First and Second Arrays :

1

2

3

4

5

6

7

8

9

10

C# Program to Search an element with Array Indices

This C# Program Searches an element with Array Indices. Here the the element is searched in the array.

Here is source code of the C# Program to Search an element with Array Indices. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Search an element with Array Indices*
3. *\*/*
4. **using** System;
6. **class** ArrayBinarySearch
7. {
8. **public** **static** **void** Main()
9. {
10. **int**[] ints = { 0, 10, 100, 1000, 1000000 };
11. Console.WriteLine("Array indices and elements: ");
12. **for** (**int** i = 0; i < ints.Length; i++)
13. {
14. Console.Write("[{0}]={1, -5}", i, ints[i]);
15. }
16. Console.WriteLine();
17. FindObject(ints, 25);
18. FindObject(ints, 1000);
19. FindObject(ints, 2000000);
20. Console.ReadLine();
21. }
23. **public** **static** **void** FindObject(Array array, **Object** o)
24. {
25. **int** index = Array.BinarySearch(array, 0, array.Length, o);
26. Console.WriteLine();
27. **if** (index > 0)
28. {
29. Console.WriteLine("Object: {0} found at [{1}]", o, index);
30. }
31. **else** **if** (~index == array.Length)
32. {
33. Console.WriteLine("Object: {0} not found. "
34. + "No array object has a greater value.", o);
35. Console.WriteLine();
36. }
37. **else**
38. {
39. Console.WriteLine("Object: {0} not found. "
40. + "Next larger object found at [{1}].", o, ~index);
41. }
42. }
43. }

advertisements

Here is the output of the C# Program:

Array indices and elements:

[0]=0 [1]=10 [2]=100 [3]=1000 [4]=1000000

Object: 25 not found. Next larger object found at [2].

Object: 1000 found at [3]

Object: 2000000 not found. No array object has a greater value.

C# Program to Implement Use of Indexers

This C# Program Implements Use of Indexers.Indexers allow instances of a class or struct to be indexed just like arrays. Indexers resemble properties except that their accessors take parameters.

Here is source code of the C# Program to Implement Use of Indexers. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Implement Use of Indexers*
3. *\*/*
4. **class** values
5. {
6. **private** **int**[] val = new **int**[10] { 10,20,30,40,50,60,70,80,90,100 };
7. **public** **int** Length
8. {
9. **get** { **return** val.Length; }
10. }
11. **public** **int** **this**[**int** index]
12. {
13. **get**
14. {
15. **return** val[index];
16. }
18. **set**
19. {
20. val[index] = **value**;
21. }
22. }
23. }
24. **class** MainClass
25. {
26. **static** **void** Main()
27. {
28. values newval = new values();
29. newval[3] = 58;
30. newval[5] = 60;
31. **for** (**int** i = 0; i < 10; i++)
32. {
33. System.Console.WriteLine("Element #{0} = {1}", i, newval[i]);
34. }
35. System.Console.WriteLine("Press any key to exit.");
36. System.Console.ReadKey();
38. }
39. }

advertisements

Here is the output of the C# Program:

Element #0 : 10

Element #1 : 20

Element #2 : 30

Element #3 : 58

Element #4 : 50

Element #5 : 60

Element #6 : 70

Element #7 : 80

Element #8 : 90

Element #9 : 100

Press any key to exit

C# Program to Find Minimum and Maximum of Numbers

This C# Program Finds Minimum and Maximum of Numbers. Here the minimum and the maximum number is obtained with the help of min and max functions.

Here is source code of the C# Program to Find Minimum and Maximum of Numbers. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Find Minimum and Maximum of Numbers*
3. *\*/*
4. **using** System;
5. **using** System.Linq;
6. **class** Program
7. {
8. **static** **void** Main()
9. {
10. **int**[] array1 = { 10, -10, -20, 0,15,20,30 };
11. Console.WriteLine("Maximum Element : " +array1.Max());
12. Console.WriteLine("Minimum Element : " +array1.Min());
13. Console.Read();
14. }
15. }

advertisements

Here is the output of the C# Program:

Maximum Element : 30

Minimum Element : -20

#### 3. C# Examples on Sorting, Searching, Copy and Reverse Operations

The programs in this section perform sorting, searching, copy and reverse operations on the elements of an array. The programs sorts a list of names alphabetically, copies a section of one array into another, searches for a given element in an array and reverses the contents of an array.

C# Program to Copy a Section of One Array to Another

This C# Program to Copy a Section of One Array to Another. Here a section of array is copied using the array.copy() with the source array,target array and the size.

Here is source code of the C# Program to Copy a Section of One Array to Another. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Copy a Section of One Array to Another*
3. *\*/*
4. **using** System;
5. **class** Program
6. {
7. **static** **void** Main()
8. {
10. **int** n, m, size;
11. Console.WriteLine("Enter the size of the Array : ");
12. n = Convert.ToInt32(Console.ReadLine());
13. **int** [] a = new **int**[n];
14. Console.WriteLine("Enter the Elements of the First Array :");
15. **for** (**int** i = 0; i <n; i++)
16. {
17. a[i] = Convert.ToInt32(Console.ReadLine());
18. }
19. Console.WriteLine("Enter the Size of the Target Array : ");
20. m = Convert.ToInt32(Console.ReadLine());
21. **int**[] target = new **int**[m];
22. Console.WriteLine("Enter the section of the First Array that has to be Copied :");
23. size = Convert.ToInt32(Console.ReadLine());
24. Array.Copy(a, 0, target, 0, size);
25. Console.WriteLine("New Array With The Specified Section of Elements in the First Array");
26. **foreach** (**int** **value** **in** target)
27. {
28. Console.WriteLine(**value**);
29. }
30. Console.Read();
31. }
32. }

advertisements

Here is the output of the C# Program:

Enter the size of the Array : 5

Enter the Elements of the First Array :

1

2

3

4

5

Enter the Size of the Target Array :

6

Enter the section of the First Array that has to be Copied : 3

New Array With The Specified Section of Elements in the First Array :

1

2

3

0

0

0

C# Program to Sort a List of Names in Alphabetical Order

This C# Program Sorts a List of Names in Alphabetical Order. Here the List of names are sorted with the help of sort function.

Here is source code of the C# Program to Sort a List of Names in Alphabetical Order. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Sort a List of Names in Alphabetical Order*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
8. **namespace** ConsoleApplication1
9. {
10. **class** Program
11. {
12. **static** **void** Main()
13. {
14. List<**string**> names = new List<**string**>();
15. names.**Add**("Ram");
16. names.**Add**("Rose");
17. names.**Add**("Abs");
18. names.**Add**("Edward");
19. names.**Add**("Sita");
20. names.Sort();
21. **foreach** (**string** s **in** names)
22. Console.WriteLine(s);
23. Console.ReadLine();
24. }
25. }
26. }

advertisements

Here is the output of the C# Program:

Abs

Edward

Ram

Rose

Sita

C# Program to Search an Element in an Array

This C# Program Searches an Element in an Array. Here an array is declared and an element is searched and if found and displayed.

Here is source code of the C# Program to Search an Element in an Array. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Search an Element in an Array*
3. *\*/*
4. **using** System;
5. **using** System.IO;
6. **class** Program
7. {
8. **static** **void** Main()
9. {
11. **string**[] array1 = { "cat", "dogs", "donkey", "camel" };
12. **string** v1 = Array.Find(array1,
13. element => element.StartsWith("cam", StringComparison.Ordinal));
14. **string** v2 = Array.Find(array1,
15. element => element.Length == 3);
16. Console.WriteLine("The Elemnt that Starts with 'Cam' is : " +v1);
17. Console.WriteLine("3 Letter word in the Array is : " +v2);
18. Console.ReadLine();
19. }
20. }

advertisements

Here is the output of the C# Program:

The Element that Starts With 'Cam' is : Camel

3 Letter Word in the Array is : cat

C# Program to Reverse an Array

This C# Program Reverses an Array. Here Array.Reverse inverts the ordering of an array’s elements.

Here is source code of the C# Program to Reverse an Array. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Reverse an Array*
3. *\*/*
4. **using** System;
5. **class** Program
6. {
7. **static** **void** Main()
8. {
9. **int**[] array = { 1, 2, 3,4,5,6,7,8,9,10 };
10. **foreach** (**int** a **in** array)
11. {
12. Console.WriteLine(a);
13. }
14. Array.Reverse(array);
15. Console.WriteLine("Reversed Array : ");
16. **foreach** (**int** **value** **in** array)
17. {
18. Console.WriteLine(**value**);
19. }
20. Console.ReadLine();
21. }
22. }

advertisements

Here is the output of the C# Program:

1

2

3

4

5

6

7

8

9

10

Reversed Array :

10

9

8

7

6

5

4

3

2

1

#### 4. C# Examples on Negation and Average Operations

The negation operator (!) negates or flips the meaning of its operand. The program in the section negates the positive elements of an array. The average of the set of values is the sum of the values divided by the total number of the values. The following program finds the average of all the elements in the array and the remaining program in the section finds the length of the jagged array using certain pre-defined functions.

C# Program to Negate the Positive Elements of Array

This C# Program Negates the Positive Elements of Array.The array elements are obtained from the user and the elements of the array are negated.

Here is source code of the C# Program to Negate the Positive Elements of Array. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Negate the Positive Elements of Array*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Text;
7. **namespace** program
8. {
9. **class** negate
10. {
11. **public** **static** **void** Main()
12. {
13. **int**[] a = new **int**[10];
14. Console.WriteLine("Enter 5 Elements : ");
15. **for** (**int** i = 0; i < 5; i++)
16. {
17. a[i] = Convert.ToInt16(Console.ReadLine());
18. **if** (a[i] > 0)
19. a[i] = -a[i];
20. }
21. Console.WriteLine("Elements:");
22. **for** (**int** i = 0; i < 5; i++)
23. {
24. Console.WriteLine(a[i]);
25. }
26. Console.ReadLine();
27. }
28. }
29. }

advertisements

Here is the output of the C# Program:

Enter 5 Elements :

10

7

8

45

67

Elements :

-10

-7

-8

-45

-67

C# Program to Find the Average Values of all the Array Elements

This C# Program Finds the Average Values of all the Array Elements. Here the array elements are obtained from the user and the sum is first calculated.Average is then found by dividing the sum by the number of terms.

Here is source code of the C# Program to Find the Average Values of all the Array Elements. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Find the Average Values of all the Array Elements*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
8. **class** program
9. {
10. **public** **void** sumAverageElements(**int**[] arr, **int** size)
11. {
13. **int** sum = 0;
14. **int** average = 0;
15. **for** (**int** i = 0; i < size; i++)
16. {
17. sum += arr[i];
18. }
19. average = sum / size;
20. Console.WriteLine("Sum Of Array is : " + sum);
21. Console.WriteLine("Average Of Array is : " + average);
22. Console.ReadLine();
23. }
24. **public** **static** **void** Main(**string**[] args)
25. {
26. **int** size;
27. Console.WriteLine("Enter the Size :");
28. size = Convert.ToInt32(Console.ReadLine());
29. **int**[] a = new **int**[size];
30. Console.WriteLine("Enter the Elements of the Array : ");
31. **for** (**int** i = 0; i < size; i++)
32. {
33. a[i] = Convert.ToInt32(Console.ReadLine());
34. }
35. **int** len = a.Length;
36. program pg = new program();
37. pg.sumAverageElements(a, len);
38. }
39. }

advertisements

Here is the output of the C# Program:

Enter the Size :

5

Enter the Elements of the Array :

10

20

30

40

50

Sum of the Array is : 150

Average of the Array is : 30

C# Program to Find the Length of the Jagged Array using Predefined Functions

This C# Program Finds the Length of the Jagged Array using Predefined Functions. Here the Length of each row in the jagged array is calculated and displayed.

Here is source code of the C# Program to Find the Length of the Jagged Array using Predefined Functions. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Find the Length of the Jagged Array using Predefined Functions*
3. *\*/*
4. **using** System;
5. **class** Program
6. {
7. **public** **static** **void** Main()
8. {
9. **byte**[][] numbers = new **byte**[5][];
10. **for** (**int** i = 0; i < numbers.Length; i++)
11. {
12. numbers[i] = new **byte**[i + 3];
13. }
14. **for** (**int** i = 0; i < numbers.Length; i++)
15. {
16. Console.WriteLine("Length of row {0} is {1}", i, numbers[i].Length);
17. }
18. Console.Read();
19. }
20. }

advertisements

Here is the output of the C# Program:

Length of row 0 is 3

Length of row 1 is 4

Length of row 2 is 5

Length of row 3 is 6

Length of row 4 is 7

#### E. C# Programming Examples on Sorting

This section has C# programs which illustrate different types of sorting operations.

#### C# Examples on Different Types of Sorting Algorithms

The following programs demonstrate different types of sorting methods. The Selection Sort finds the smallest element in the array and exchanges it with the element in the first position, it then finds the second smallest element and exchanges it with the element in the second position and continues this process until the entire list is sorted. LSDRadix Algorithm is used to sort the keys in integer representation order. Heap Sort is based on binary heap data structure and sorts elements similar to selection sort where the maximum element is placed at the end of the list. The Merge Sort follows the Divide and Conquer rule where in it divides the input array into two halves, sorts the two halves and merges the two sorted halves. The Quick Sort selects an element as a pivot and partitions the given array around the pivot. Bubble Sort repeatedly sorts the adjacent elements if they are in wrong order.

C# Program to Perform Radix Sort

This C# Program Performs Radix Sort.Radix Sort is a non-comparative integer sorting algorithm that sorts data with integer keys by grouping keys by the individual digits which share the same significant position and value

Here is source code of the C# program which performs Radix Sort. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Perform Radix Sort*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
9. **namespace** ConsoleApplication2
10. {
11. **class** Example
12. {
13. **private** **int**[] data;
14. **private** IList<IList<**int**>> digits = new List<IList<**int**>>();
15. **private** **int** maxLength = 0;
16. **public** Example()
17. {
18. **for** (**int** i = 0; i < 10; i++)
19. {
20. digits.**Add**(new List<**int**>());
21. }
22. Console.Write("Enter the Number of Records : ");
23. **int** count = **int**.Parse(Console.ReadLine());
24. data = new **int**[count];
25. Console.ReadLine();
26. **for** (**int** i = 0; i < count; i++)
27. {
28. Console.Write("Enter Record {0} : ", i + 1);
30. data[i] = **int**.Parse(Console.ReadLine());
32. **if** (maxLength < data[i].ToString().Length)
33. maxLength = data[i].ToString().Length;
34. }
35. }
37. **public** **void** RadixSort()
38. {
39. **for** (**int** i = 0; i < maxLength; i++)
40. {
41. **for** (**int** j = 0; j < data.Length; j++)
42. {
43. **int** digit = (**int**)((data[j] % Math.Pow(10, i + 1)) / Math.Pow(10, i));
45. digits[digit].**Add**(data[j]);
46. }
48. **int** index = 0;
49. **for** (**int** k = 0; k < digits.Count; k++)
50. {
51. IList<**int**> selDigit = digits[k];
53. **for** (**int** l = 0; l < selDigit.Count; l++)
54. {
55. data[index++] = selDigit[l];
56. }
57. }
58. ClearDigits();
59. }
60. printSortedData();
61. }
63. **private** **void** ClearDigits()
64. {
65. **for** (**int** k = 0; k < digits.Count; k++)
66. {
67. digits[k].Clear();
68. }
69. }
71. **public** **void** printSortedData()
72. {
73. Console.WriteLine("The Sorted Numbers are : ");
74. **for** (**int** i = 0; i < data.Length; i++)
75. {
76. Console.WriteLine(data[i]);
77. }
78. }
79. **static** **void** Main(**string**[] args)
80. {
81. new Example().RadixSort();
83. Console.ReadLine();
84. }
85. }
86. }

advertisements

Here is the output of the C# Program:

Enter the Number of Records : 5

Enter Record 1 : 54

Enter Record 2 : 53

Enter Record 3 : 15

Enter Record 4 : 27

Enter Record 5 : 75

The Sorted Numbers are :

15

27

53

54

75

C# Program to Perform a Selection Sort

This C# Program Performs a Selection Sort.Here Selection sort is an algorithm of sorting an array where it loop from the start of the loop, and check through other elements to find the minimum value. After the end of the first iteration, the minimum value is swapped with the current element. The iteration then continues from the 2nd element and so on.

Here is source code of the C# Program to Perform a Selection Sort. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Perform a Selection Sort*
3. *\*/*
4. **using** System;
5. **class** Program
6. {
7. **static** **void** Main(**string**[] args)
8. {
9. **int** array\_size = 10;
10. **int**[] array = new **int**[10] { 100, 50, 20, 40, 10, 60, 80, 70, 90, 30 };
11. Console.WriteLine("The Array Before Selection Sort is: ");
12. **for** (**int** i = 0; i < array\_size; i++)
13. {
14. Console.WriteLine(array[i]);
15. }
16. **int** tmp, min\_key;
18. **for** (**int** j = 0; j < array\_size - 1; j++)
19. {
20. min\_key = j;
22. **for** (**int** k = j + 1; k < array\_size; k++)
23. {
24. **if** (array[k] < array[min\_key])
25. {
26. min\_key = k;
27. }
28. }
30. tmp = array[min\_key];
31. array[min\_key] = array[j];
32. array[j] = tmp;
33. }
35. Console.WriteLine("The Array After Selection Sort is: ");
36. **for** (**int** i = 0; i < 10; i++)
37. {
38. Console.WriteLine(array[i]);
39. }
40. Console.ReadLine();
41. }
42. }

advertisements

Here is the output of the C# Program:

The Array Before Selection Sort is :

100

50

20

40

10

60

80

70

90

30

The Array After Selection Sort is :

10

20

30

40

50

60

70

80

90

100

C# Program to Demonstrate Heap Sort

This C# Program Demonstrates Heap Sort. Here it first removes the topmost item (the largest) and replace it with the rightmost leaf. The topmost item is stored in an array and Re-establish the heap.this is done until there are no more items left in the heap.

Here is source code of the C# Program to Demonstrate Heap Sort. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Demonstrate Heap Sort*
3. *\*/*
4. **using** System;
5. **class** heap
6. {
7. **int**[] r = { 2,5,1,10,6,9,3,7,4,8};
8. **public** **void** hsort()
9. {
10. **int** i, t;
11. **for** (i = 5; i >= 0; i--)
12. {
13. adjust(i, 9);
14. }
15. **for** (i = 8; i >= 0; i--)
16. {
17. t = r[i + 1];
18. r[i + 1] = r[0];
19. r[0] = t;
20. adjust(0, i);
21. }
22. }
23. **private** **void** adjust(**int** i, **int** n)
24. {
25. **int** t, j;
26. **try**
27. {
28. t = r[i];
29. j = 2 \* i;
30. **while** (j <= n)
31. {
32. **if** (j < n && r[j] < r[j + 1])
33. j++;
34. **if** (t >=r[j])
35. **break**;
36. r[j / 2] = r[j];
37. j \*= 2;
38. }
39. r[j / 2] = t;
40. }
41. **catch** (IndexOutOfRangeException e)
42. {
43. Console.WriteLine("Array Out of Bounds ", e);
44. }
45. }
46. **public** **void** print()
47. {
48. **for** (**int** i = 0; i < 10; i++)
49. {
50. Console.WriteLine("{0}", r[i]);
51. }
53. }
54. **public** **static** **void** Main()
55. {
56. heap obj = new heap();
57. Console.WriteLine("Elements Before sorting : ");
58. obj.print();
59. obj.hsort();
60. Console.WriteLine("Elements After sorting : ");
61. obj.print();
62. Console.Read();
63. }
64. }

advertisements

Here is the output of the C# Program:

Elements Before Sorting :

2

5

1

10

6

9

3

7

4

8

Elements After Sorting :

1

2

3

4

5

6

7

8

9

10

C# Program to Perform Bubble Sort

This C# Program Performs Bubble Sort. Here bubble sort changes the postion of numbers or changing an unordered sequence into an ordered sequence.

Here is source code of the C# Program to Perform Bubble Sort. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Perform Bubble Sort*
3. *\*/*
4. **using** System;
5. **class** bubblesort
6. {
7. **static** **void** Main(**string**[] args)
8. {
9. **int**[] a = { 3, 2, 5, 4, 1 };
10. **int** t;
11. Console.WriteLine("The Array is : ");
12. **for** (**int** i = 0; i < a.Length; i++)
13. {
14. Console.WriteLine(a[i]);
15. }
16. **for** (**int** j = 0; j <= a.Length - 2; j++)
17. {
18. **for** (**int** i = 0; i <= a.Length - 2; i++)
19. {
20. **if** (a[i] > a[i + 1])
21. {
22. t = a[i + 1];
23. a[i + 1] = a[i];
24. a[i] = t;
25. }
26. }
27. }
28. Console.WriteLine("The Sorted Array :");
29. **foreach** (**int** aray **in** a)
30. Console.Write(aray + " ");
31. Console.ReadLine();
32. }
33. }

advertisements

Here is the output of the C# Program:

The Array is :

3

2

5

4

1

The Sorted Array :

1

2

3

4

5

C# Program to Perform Insertion Sort

This C# Program Performs Insertion Sort. Here it takes an element from the list and places it in the correct location in the list. This process is repeated until there are no more unsorted items in the list.

Here is source code of the C# Program to Perform Insertion Sort. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Perform Insertion Sort*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
8. **using** System.IO;
9. **namespace** ConsoleApplication1
10. {
11. **class** Program
12. {
13. **static** **void** Main(**string**[] args)
14. {
15. **int**[] arr = new **int**[5] { 83, 12, 3, 34, 60 };
16. **int** i;
17. Console.WriteLine("The Array is :");
18. **for** (i = 0; i < 5; i++)
19. {
20. Console.WriteLine(arr[i]);
21. }
22. insertsort(arr, 5);
23. Console.WriteLine("The Sorted Array is :");
24. **for** (i = 0; i < 5; i++)
25. Console.WriteLine(arr[i]);
26. Console.ReadLine();
27. }
28. **static** **void** insertsort(**int**[] data, **int** n)
29. {
30. **int** i, j;
31. **for** (i = 1; i < n; i++)
32. {
33. **int** item = data[i];
34. **int** ins = 0;
35. **for** (j = i - 1; j >= 0 && ins != 1; )
36. {
37. **if** (item < data[j])
38. {
39. data[j + 1] = data[j];
40. j--;
41. data[j + 1] = item;
42. }
43. **else** ins = 1;
44. }
45. }
46. }
47. }
48. }

advertisements

Here is the output of the C# Program:

The Array is :

83

12

3

34

60

The Sorted Array is :

3

12

34

60

83

C# Program to Perform Merge Sort

This C# Program Performs Merge Sort. A merge sort is a sorting algorithm with complexity of O(nlogn). It is used for sorting numbers, structure, files.

Here is source code of the C# Program to Perform Merge Sort. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Perform Merge Sort*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Text;
7. **namespace** prog
8. {
9. **class** Program
10. {
11. **static** **public** **void** mergemethod(**int** [] numbers, **int** left, **int** mid, **int** right)
12. {
13. **int** [] temp = new **int**[25];
14. **int** i, left\_end, num\_elements, tmp\_pos;
15. left\_end = (mid - 1);
16. tmp\_pos = left;
17. num\_elements = (right - left + 1);
18. **while** ((left <= left\_end) && (mid <= right))
19. {
20. **if** (numbers[left] <= numbers[mid])
21. temp[tmp\_pos++] = numbers[left++];
22. **else**
23. temp[tmp\_pos++] = numbers[mid++];
24. }
25. **while** (left <= left\_end)
26. temp[tmp\_pos++] = numbers[left++];
27. **while** (mid <= right)
28. temp[tmp\_pos++] = numbers[mid++];
29. **for** (i = 0; i < num\_elements; i++)
30. {
31. numbers[right] = temp[right];
32. right--;
33. }
35. }
36. **static** **public** **void** sortmethod(**int** [] numbers, **int** left, **int** right)
37. {
38. **int** mid;
39. **if** (right > left)
40. {
41. mid = (right + left) / 2;
42. sortmethod(numbers, left, mid);
43. sortmethod(numbers, (mid + 1), right);
44. mergemethod(numbers, left, (mid+1), right);
46. }
47. }
48. **static** **void** Main(**string**[] args)
50. {
52. **int**[] numbers = { 3, 8, 7, 5, 2, 1, 9, 6, 4 };
53. **int** len = 9;
54. Console.WriteLine("MergeSort :");
55. sortmethod(numbers, 0, len - 1);
56. **for** (**int** i = 0; i < 9; i++)
57. Console.WriteLine(numbers[i]);
58. Console.Read();
59. }
60. }
61. }

advertisements

Here is the output of the C# Program:

MergeSort :

1

2

3

4

5

6

7

8

9

C# Program to Implement Quick Sort

This C# Program Implements Quick Sort.Quicksort is a divide and conquer algorithm. Here Quicksort first divides a large array into two smaller sub-array: the low elements and the high elements. Quicksort can then recursively sort the sub-arrays

Here is source code of the C# Program to Implement Quick Sort. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Implement Quick Sort*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
9. **namespace** sortQuick
10. {
11. **class** quickSort
12. {
14. **private** **int**[] array = new **int**[20];
15. **private** **int** len;
17. **public** **void** QuickSort()
18. {
19. sort(0, len - 1);
20. }
22. **public** **void** sort(**int** left, **int** right)
23. {
24. **int** pivot, leftend, rightend;
26. leftend = left;
27. rightend = right;
28. pivot = array[left];
30. **while** (left < right)
31. {
32. **while** ((array[right] >= pivot) && (left < right))
33. {
34. right--;
35. }
37. **if** (left != right)
38. {
39. array[left] = array[right];
40. left++;
41. }
43. **while** ((array[left] <= pivot) && (left < right))
44. {
45. left++;
46. }
48. **if** (left != right)
49. {
50. array[right] = array[left];
51. right--;
52. }
53. }
55. array[left] = pivot;
56. pivot = left;
57. left = leftend;
58. right = rightend;
60. **if** (left < pivot)
61. {
62. sort(left, pivot - 1);
63. }
65. **if** (right > pivot)
66. {
67. sort(pivot + 1, right);
68. }
69. }
71. **public** **static** **void** Main()
72. {
73. quickSort q\_Sort = new quickSort();
75. **int**[] array = { 4, 3, 1, 4, 6, 7, 5, 4, 32, 5, 26, 187, 8 };
76. q\_Sort.array = array;
77. q\_Sort.len = q\_Sort.array.Length;
78. q\_Sort.QuickSort();
80. **for** (**int** j = 0; j < q\_Sort.len; j++)
81. {
82. Console.WriteLine(q\_Sort.array[j]);
83. }
84. Console.ReadKey();
85. }
86. }
87. }

advertisements

Here is the output of the C# Program:

1

3

4

4

4

5

5

6

7

8

26

32

187

#### F. C# Programming Examples on Conversions

The programs in this section deal with various types of conversions. These are number system conversions, decimal and hexadecimal notations, prefix and postfix, big endian and little endian notations.

#### 1. C# Examples on Number System Conversions

A Binary System has two possible values 0 and 1, Decimal System has 10 possible values (0,1,2,3,4,5,6,7,8,9) and a Octal System has 8 possible values (0,1,2,3,4,5,6,7). The programs in the section perform Binary to Decimal conversion, Decimal to Binary conversion and Decimal to Octal conversion. It also converts temperature in the Celsius scale to the temperature in the Fahrenheit scale. A Gray code is the number encoding such that adjacent numbers have a single digit differing by 1. The program in the section creates a gray code, gets the character from the user and converts the case of the character.

C# Program to Perform Binary to Decimal Conversion

This C# Program Performs Binary to Decimal Conversion. This C# Program converts the given binary number into decimal. The program reads the binary number, does a modulo operation to get the remainder, multiples the total by base 2 and adds the modulo and repeats the steps.

Here is source code of the C# Program to Perform Binary to Decimal Conversion.The C# program is successfully compiled and executed with Microsoft Visual Studio.The program output is also shown below.

1. */\**
2. *\* C# Program to Perform Binary to Decimal Conversion*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Text;
8. **namespace** Program
9. {
10. **class** Program
11. {
12. **static** **void** Main(**string**[] args)
13. {
14. **int** num, binary\_val, decimal\_val = 0, base\_val = 1, rem;
15. Console.Write("Enter a Binary Number(1s and 0s) : ");
16. num = **int**.Parse(Console.ReadLine()); */\* maximum five digits \*/*
17. binary\_val = num;
18. **while** (num > 0)
19. {
20. rem = num % 10;
21. decimal\_val = decimal\_val + rem \* base\_val;
22. num = num / 10 ;
23. base\_val = base\_val \* 2;
24. }
25. Console.Write("The Binary Number is : "+binary\_val);
26. Console.Write("**\n**Its Decimal Equivalent is : "+decimal\_val);
27. Console.ReadLine();
28. }
29. }
30. }

advertisements

Here is the output of the C# Program:

Enter a Binary Number(1s and 0s) : 101010

The Binary Number is : 101010

Its Decimal Equivalent is : 42

C# Program to Convert Decimal to Binary

This C# Program Converts Decimal to Binary. Here the number in decimal form is obtained and is it repeatedly divided by 2 and its binary form is obtained.

Here is source code of the C# Program to Convert Decimal to Binary. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Convert Decimal to Binary*
3. *\*/*
4. **using** System;
5. **class** myclass
6. {
7. **static** **void** Main()
8. {
9. **int** num;
10. Console.Write("Enter a Number : ");
11. num = **int**.Parse(Console.ReadLine());
12. **int** quot;
13. **string** rem = "";
14. **while** (num >= 1)
15. {
16. quot = num / 2;
17. rem += (num % 2).ToString();
18. num = quot;
19. }
20. **string** bin = "";
21. **for** (**int** i = rem.Length - 1; i >= 0; i--)
22. {
23. bin = bin + rem[i];
24. }
25. Console.WriteLine("The Binary format for given number is {0}", bin);
26. Console.Read();
27. }
28. }

advertisements

Here is the output of the C# Program:

Enter the Number : 3

Binary Format for the Given Number is : 11

C# Program to Convert Decimal to Octal

This C# Program Converts Decimal to Octal. Here the decimal number is first obtained from the user and Is converted to octal.

Here is source code of the C# Program to Convert Decimal to Octal. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Convert Decimal to Octal*
3. *\*/*
4. **using** System;
5. **class** program
6. {
7. **public** **static** **void** Main()
8. {
9. **int** decimalNumber, quotient, i = 1, j;
10. **int**[] octalNumber = new **int**[100];
11. Console.WriteLine("Enter a Decimal Number :");
12. decimalNumber = **int**.Parse(Console.ReadLine());
13. quotient = decimalNumber;
14. **while** (quotient != 0)
15. {
16. octalNumber[i++] = quotient % 8;
17. quotient = quotient / 8;
18. }
19. Console.Write("Equivalent Octal Number is ");
20. **for** (j = i - 1; j > 0; j--)
21. Console.Write(octalNumber[j]);
22. Console.Read();
23. }
24. }

advertisements

Here is the output of the C# Program:

Enter a Decimal Number :

50

Equivalent Octal Number is 62

C# Program to Perform Celsius to Fahrenheit Conversion

This C# Program Performs Celsius to Fahrenheit Conversion. Here the temperature is obtained in Celsius and is converted to Fahrenheit .

Here is source code of the C# program to perform Celsius to Fahrenheit conversion. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Perform Celsius to Fahrenheit Conversion*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
9. **namespace** program
10. {
11. **class** Program
12. {
13. **static** **void** Main(**string**[] args)
14. {
15. **int** celsius, faren;
16. Console.WriteLine("Enter the Temperature in Celsius(°C) : ");
17. celsius = **int**.Parse(Console.ReadLine());
18. faren = (celsius \* 9) / 5 + 32;
19. Console.WriteLine("0Temperature in Fahrenheit is(°F) : " + faren);
20. Console.ReadLine();
22. }
23. }
24. }

advertisements

Here is the output of the C# Program:

Enter the Temperature in Celsius (°C) : 20

Temperature in Fahrenheit(°F) is : 68

C# Program to Convert Fahrenheit to Celsius

This C# Program Converts Fahrenheit to Celsius. Here To convert from Fahrenheit to Celsius, first subtract 32, then multiply by 100/180.

Here is source code of the C# Program to Convert Fahrenheit to Celsius. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Convert Fahrenheit to Celsius*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
9. **namespace** Celsius
10. {
11. **class** Program
12. {
13. **static** **void** Main(**string**[] args)
14. {
15. **double** celsius;
16. Console.Write("Enter Fahrenheit temperature : ");
17. **double** fahrenheit = Convert.ToDouble(Console.ReadLine());
18. celsius = (fahrenheit - 32) \* 5 / 9;
19. Console.WriteLine("The converted Celsius temperature is" + celsius);
20. Console.ReadLine();
21. }
22. }
23. }

advertisements

Here is the output of the C# Program:

Output:

Enter Fahrenheit temperature : 95.5

The converted Celsius temperature is35.2777777777778

C# Program to Create a Gray Code

This C# Program Creates a Gray Code. A Gray code is an encoding of numbers so that adjacent numbers have a single digit differing by 1. The term Gray code is often used to refer to a “reflected” code, or more specifically still, the binary reflected Gray code.

Here is source code of the C# Program to Create a Gray Code. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Create a Gray Code*
3. *\*/*
4. **using** System;
5. **public** **class** Gray
6. {
7. **public** **static** **ulong** grayEncode(**ulong** n)
8. {
9. **return** n ^ (n >> 1);
10. }
12. **public** **static** **ulong** grayDecode(**ulong** n)
13. {
14. **ulong** i = 1 << 8 \* 64 - 2; *//long is 64-bit*
15. **ulong** p, b = p = n & i;
17. **while** ((i >>= 1) > 0)
18. b |= p = n & i ^ p >> 1;
19. **return** b;
20. }
22. **public** **static** **void** Main(**string**[] args)
23. {
24. Console.WriteLine("Number**\t**Gray");
25. **for** (**ulong** i = 0; i < 10; i++)
26. {
27. Console.WriteLine(**string**.Format("{0}**\t**{1}", i, Convert.ToString((**long**)grayEncode(i), 2)));
29. }
30. Console.Read();
31. }
32. }

advertisements

Here is the output of the C# Program:

Number Gray

0 0

1 1

2 11

3 10

4 110

5 111

6 101

7 100

8 1100

9 1101

C# Program to Obtain the Character from the User and Convert the Case of the Character

This C# Program Obtains the Character from the User and Convert the Case of the Character. Here the user enters the character and the uppercase letters are converted to lowercase and vice versa.

Here is source code of the C# Program to Obtain the Character from the User and Convert the Case of the Character. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Obtain the Character from the User and Convert the Case of the Character*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
8. **namespace** casechange
9. {
10. **class** Program
11. {
12. **static** **void** Main(**string**[] args)
13. {
14. **char** a;
15. **int** i;
16. Console.WriteLine("Enter the Character : ");
17. a = Convert.ToChar(Console.ReadLine());
18. i=(**int**)a;
19. **if** (a >= 65 && a <= 90)
20. {
22. Console.WriteLine("The Character is : {0}", **char**.ToLower(a));
24. }
25. **else** **if** (a >= 97 && a <= 122)
26. {
27. Console.WriteLine("The Character is : {0}", **char**.ToUpper(a));
28. }
29. Console.ReadLine();
30. }
31. }

advertisements

Here is the output of the C# Program:

Enter the Character : a

The Character is : A

#### 2. C# Examples Dealing with Hexadecimal and Decimal Notations

The digits used to represent numbers using hexadecimal notation are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F. The following programs convert a hexadecimal number into a decimal number and vice versa. The programs also convert meter to kilometer and vice versa. The program also converts a given number of days in terms of years, weeks & days. The remaining program in the section performs various type conversions.

C# Program to Convert a Given Number of Days in terms of Years, Weeks & Days

This C# Program Converts a Given Number of Days in terms of Years, Weeks & Days. Here This program accepts the number of days. Given the number of days, then it calculates the years, weeks & days for this number.

Here is source code of the C# Program to Convert a Given Number of Days in terms of Years, Weeks & Days. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Convert a Given Number of Days in terms of*
3. *\* Years, Weeks & Days*
4. *\*/*
5. **using** System;
6. **class** program
7. {
8. **public** **static** **void** Main()
9. {
10. **int** ndays, year, week, days, DAYSINWEEK=7;
11. Console.WriteLine("Enter the number of days");
12. ndays = **int**.Parse(Console.ReadLine());
13. year = ndays / 365;
14. week = (ndays % 365) / DAYSINWEEK;
15. days = (ndays % 365) % DAYSINWEEK;
16. Console.WriteLine("{0} is equivalent to {1}years, {2}weeks and {3}days", ndays, year, week, days);
17. Console.ReadLine();
18. }
19. }

advertisements

Here is the output of the C# Program:

Enter the number of days

1000

1000 is equivalent to 2 years, 38 weeks and 4 days

C# Program to Perfom Currency Conversions

This C# Program to Perfom Currency Conversions. Here the currency conversions are made based on the choice that is given by the user and the conversions are made based on the exchange value and the corresponding value is displayed.

Here is source code of the C# Program to Perfom Currency Conversions. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Perfom Currency Conversions*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
8. **namespace** Program
9. {
10. **class** Program
11. {
12. **static** **void** Main(**string**[] args)
13. {
14. **int** choice;
15. Console.WriteLine("Enter your Choice :**\n** 1- Dollar to Rupee **\n** 2- Euro to Rupee **\n** 3- Malaysian Ringgit to Rupee ");
16. choice = **int**.Parse(Console.ReadLine());
17. **switch** (choice)
18. {
19. **case** 1:
20. **Double** dollar, rupee,val;
21. Console.WriteLine("Enter the Dollar Amount :");
22. dollar = **Double**.Parse(Console.ReadLine());
23. Console.WriteLine("Enter the Dollar Value :");
24. val = **double**.Parse(Console.ReadLine());
25. rupee = dollar \* val;
26. Console.WriteLine("{0} Dollar Equals {1} Rupees", dollar, rupee);
27. **break**;
28. **case** 2:
29. **Double** Euro, rupe,valu;
30. Console.WriteLine("Enter the Euro Amount :");
31. Euro = **Double**.Parse(Console.ReadLine());
32. Console.WriteLine("Enter the Euro Value :");
33. valu = **double**.Parse(Console.ReadLine());
34. rupe = Euro \* valu;
35. Console.WriteLine("{0} Euro Equals {1} Rupees", Euro, rupe);
36. **break**;
37. **case** 3:
38. **Double** ringit, rup,**value**;
39. Console.WriteLine("Enter the Ringgit Amount :");
40. ringit = **Double**.Parse(Console.ReadLine());
41. Console.WriteLine("Enter the Ringgit Value :");
42. **value** = **double**.Parse(Console.ReadLine());
43. rup = ringit \* **value**;
44. Console.WriteLine("{0} Malaysian Ringgit Equals {1} Rupees", ringit, rup);
45. **break**;
46. }
47. Console.ReadLine();
48. }
49. }
50. }

advertisements

Here is the output of the C# Program:

Enter the Choice :

1 - Dollar to Rupee

2 - Euro to Rupee

3 - Malaysian Ringgit to Rupee

1

Enter the Dollar Amount : 20

Enter the Dollar Value : 62.58

20 Dollar Equals 1251.6 Rupees

C# Program to Covert HexaDecimal Number to Decimal

This C# Program to Covert HexaDecimal Number to Decimal. Here the hexadecimal number is obtained from the user and its decimal equivalent.

Here is source code of the C# Program to Covert HexaDecimal Number to Decimal. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Covert HexaDecimal Number to Decimal*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
9. **namespace** ConsoleApplication
10. {
11. **class** Program
12. {
13. **static** **void** Main(**string**[] args)
14. {
15. **string** Input;
16. Console.WriteLine("Enter a Hexadecimal Number :");
17. Input = Console.ReadLine();
18. **int** Output = **int**.Parse(Input, System.Globalization.NumberStyles.HexNumber);
19. Console.WriteLine("The Decimal value is " + Output);
20. Console.Read();
21. }
22. }
23. }

advertisements

Here is the output of the C# Program:

Enter a HexaDecimal Number : ABC

The Decimal value is : 2748

C# Program to Perform Decimal to HexaDecimal Conversion

This C# Program Performs Decimal to HexaDecimal Conversion. Here the decimal number is first obtained from the user and Is converted to hexadecimal.

Here is source code of the C# Program to Perform Decimal to HexaDecimal Conversion. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Perform Decimal to HexaDecimal Conversion*
3. *\*/*
4. **using** System;
5. **class** program
6. {
7. **public** **static** **void** Main()
8. {
9. **int** decimalNumber, quotient;
10. **int** i = 1, j, temp = 0;
11. **char**[] hexadecimalNumber = new **char**[100];
12. **char** temp1;
13. Console.WriteLine("Enter a Decimal Number :");
14. decimalNumber = **int**.Parse(Console.ReadLine());
15. quotient = decimalNumber;
16. **while** (quotient != 0)
17. {
18. temp = quotient % 16;
19. **if** (temp < 10)
20. temp = temp + 48;
21. **else**
22. temp = temp + 55;
23. temp1 = Convert.ToChar(temp);
24. hexadecimalNumber[i++] = temp1;
25. quotient = quotient / 16;
26. }
27. Console.Write("Equivalent HexaDecimal Number is ");
28. **for** (j = i - 1; j > 0; j--)
29. Console.Write(hexadecimalNumber[j]);
30. Console.Read();
32. }
33. }

advertisements

Here is the output of the C# Program:

Enter a Decimal Number :

45

Equivalent HexaDecimal Number is 2D

C# Program to Perform Conversions of Meter to Kilometer and viceversa

This C# Program Performs Conversions of Meter to Kilometer and viceversa. Here the entered meter value is converted into kilometer and the entered kilometer is converted interms of meter.

Here is source code of the C# Program to Perform Conversions of Meter to Kilometer and viceversa. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Perform Conversions of Meter to Kilometer and viceversa*
3. *\*/*
4. **using** System;
5. **class** Program
6. {
7. **static** **void** Main()
8. {
9. **double** m1 = 200;
10. Console.WriteLine("Meter :: Kilometer");
11. **double** k1 = ConvertDistance.cMtK(100);
12. Console.WriteLine("{0} :: {1}", m1, k1);
13. **double** m4 = 310.7;
14. **double** k4 = ConvertDistance.cMtK(4200.7);
15. Console.WriteLine("{0} :: {1}", m4, k4);
16. **double** k3 = 500;
17. Console.WriteLine();
18. Console.WriteLine("Kilometer :: Meter");
19. **double** m3 = ConvertDistance.cKtM(500);
20. Console.WriteLine("{0} :: {1}", k3, m3);
21. **double** k2 = 321.9;
22. **double** m2 = ConvertDistance.cKtM(451.9);
23. Console.WriteLine("{0} :: {1}", k2, m2);
24. Console.Read();
25. }
26. }
28. **public** **static** **class** ConvertDistance
29. {
30. **public** **static** **double** cMtK(**double** miles)
31. {
32. **return** miles \* 1.609344;
33. }
35. **public** **static** **double** cKtM(**double** kilometers)
36. {
37. **return** kilometers \* 0.621371192;
38. }
39. }

advertisements

Here is the output of the C# Program:

Meter :: Kilometer

200 :: 160.9344

310.7 :: 6760.3713408

Kilometer :: Meter

500 :: 310.685596

321.9 :: 280.7976416648

#### 3. C# Examples dealing with Prefix, Postfix, Big Endian and Little Endian Notation

Operators are written in-between their operands in Infix notation and Operators are written after their operands in Postfix notation. Big-endian is an order in which the most significant value is stored first. Little-endian is an order in which the least significant value is stored first. The programs in the following section converts an Infix expression to a Postfix expression, converts Big Endian format to Little Endian format. The remaining programs in the section converts digits to words and demonstrates DefaultIfEmpty case.

C# Program to Convert Digits to Words

This C# Program Converts Digits to Words. Here the user enters the number which is again converted and diplayed in terms of words with the help of a mod function.

Here is source code of the C# Program to Convert Digits to Words. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Convert Digits to Words*
3. *\*/*
4. **using** System;
6. **public** **class** ConvertDigitsToWords
7. {
8. **public** **static** **void** Main()
9. {
10. **int** num;
11. **int** nextdigit;
12. **int** numdigits;
13. **int**[] n = new **int**[20];
15. **string**[] digits = { "zero", "one", "two",
16. "three", "four", "five",
17. "six", "seven", "eight",
18. "nine" };

21. Console.WriteLine("Enter the number");
22. num = Convert.ToInt32(Console.ReadLine());
23. Console.WriteLine("Number: " + num);
24. Console.Write("Number in words: ");
25. nextdigit = 0;
26. numdigits = 0;
27. **do**
28. {
29. nextdigit = num % 10;
30. n[numdigits] = nextdigit;
31. numdigits++;
32. num = num / 10;
33. } **while**(num > 0);
34. numdigits--;
35. **for**( ; numdigits >= 0; numdigits--)
36. Console.Write(digits[n[numdigits]] + " ");
37. Console.WriteLine();
38. Console.ReadLine();
39. }
40. }

advertisements

Here is the output of the C# Program:

Enter the number

1548

Number: 1548

Number in words: one five four eight

C# Program to Convert Infix to Postfix

This C# Program Converts Infix to Postfix. Here the infix expression is obtained from the user and is converted to postfix expression which consists of primary expressions or expressions in which postfix operators follow a primary expression.

Here is source code of the C# Program to Convert Infix to Postfix. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Convert Infix to Postfix*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
8. **namespace** Infix
9. {
10. **class** Program
11. {
12. **static** **bool** convert(**ref** **string** infix, **out** **string** postfix)
13. {
15. **int** prio = 0;
16. postfix = "";
17. Stack<**Char**> s1 = new Stack<**char**>();
18. **for** (**int** i = 0; i < infix.Length; i++)
19. {
20. **char** ch = infix[i];
21. **if** (ch == '+' || ch == '-' || ch == '\*' || ch == '/')
22. {
23. **if** (s1.Count <= 0)
24. s1.Push(ch);
25. **else**
26. {
27. **if** (s1.Peek() == '\*' || s1.Peek() == '/')
28. prio = 1;
29. **else**
30. prio = 0;
31. **if** (prio == 1)
32. {
33. **if** (ch == '+' || ch == '-')
34. {
35. postfix += s1.Pop();
36. i--;
37. }
38. **else**
39. {
40. postfix += s1.Pop();
41. i--;
42. }
43. }
44. **else**
45. {
46. **if** (ch == '+' || ch == '-')
47. {
48. postfix += s1.Pop();
49. s1.Push(ch);
51. }
52. **else**
53. s1.Push(ch);
54. }
55. }
56. }
57. **else**
58. {
59. postfix += ch;
60. }
61. }
62. **int** len = s1.Count;
63. **for** (**int** j = 0; j < len; j++)
64. postfix += s1.Pop();
65. **return** **true**;
66. }
67. **static** **void** Main(**string**[] args)
68. {
69. **string** infix = "";
70. **string** postfix = "";
71. **if** (args.Length == 1)
72. {
73. infix = args[0];
74. convert(**ref** infix, **out** postfix);
75. System.Console.WriteLine("InFix :**\t**" + infix);
76. System.Console.WriteLine("PostFix:**\t**" + postfix);
77. }
78. **else**
79. {
80. infix = "a+b\*c-d";
81. convert(**ref** infix, **out** postfix);
82. System.Console.WriteLine("InFix :**\t**" + infix);
83. System.Console.WriteLine("PostFix :**\t**" + postfix);
84. System.Console.WriteLine();
85. infix = "a+b\*c-d/e\*f";
86. convert(**ref** infix, **out** postfix);
87. System.Console.WriteLine("InFix :**\t**" + infix);
88. System.Console.WriteLine("PostFix :**\t**" + postfix);
89. System.Console.WriteLine();
90. infix = "a-b/c\*d-e--f/h\*i++j-/k";
91. convert(**ref** infix, **out** postfix);
92. System.Console.WriteLine("InFix :**\t**" + infix);
93. System.Console.WriteLine("PostFix :**\t**" + postfix);
94. System.Console.WriteLine();
95. Console.ReadLine();
96. }
97. }
98. }
99. }

advertisements

Here is the output of the C# Program:

Infix : a+b\*c-d

Postfix : abc\*+d-

Infix : a+b\*c-d/e\*f

Postfix : abc\*+de/f\*-

Infix : a-b/c\*d-e--f/h\*I++j-/k

Postfix : abc/d\*-e--fh/I\*-=j=k/-

C# Program to Demonstrate DefaultIfEmpty case

This C# Program Demonstrates DefaultIfEmpty case. Here the default ifempty case is explained with the student database.

Here is source code of the C# Program to Demonstrate DefaultIfEmpty casee. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Demonstrate DefaultIfEmpty case*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
9. **namespace** program
10. {
11. **class** student
12. {
13. **public** **string** Name { **get**; **set**; }
14. **public** **string** stuID { **get**; **set**; }
16. }
17. **class** studentdetails
18. {
19. **public** **string** SID { **get**; **set**; }
20. **public** **string** City { **get**; **set**; }
21. }
23. **class** Program
24. {
25. **static** **void** Main(**string**[] args)
26. {
27. List<student> objstudent = new List<student>{
28. new student{ Name="Bob",stuID="I001"},
29. new student{ Name="Vijay",stuID="I002"},
30. new student{ Name="Jerry",stuID="I003"},
31. new student{ Name="Tom",stuID="I004"},
32. new student{ Name="Senthil",stuID="I005"},
33. };
34. List<studentdetails> objstudentdetails = new List<studentdetails>{
35. new studentdetails{ SID="I001",City="Delhi"},
36. new studentdetails{ SID="I002",City="Mumbai"},
37. new studentdetails{ SID="I007",City="Chennai"},
38. new studentdetails{ SID="I008",City="Pune"},
39. new studentdetails{ SID="I009",City=""},
40. };
41. **var** resultDefaultIfEmpty = **from** stu **in** objstudent
42. **join** studentdetails **in** objstudentdetails on stu.stuID equals studentdetails.SID **into** ResultEmpstudentdetails
44. **from** output **in** ResultEmpstudentdetails.DefaultIfEmpty()
45. **select** new
46. {
47. studentName = stu.Name,
48. City = output != **null** ? output.City : **null**
49. };
50. Console.WriteLine(**string**.**Join**("**\n**", resultDefaultIfEmpty.**Select**(stu => " student Name = " +
51. stu.studentName + ", City Name = " + stu.City).ToArray<**string**>()));
52. Console.ReadLine();
53. }
54. }
55. }

advertisements

Here is the output of the C# Program:

student Name = Bob, City Name = Delhi

student Name = Vijay, City Name = Mumbai

student Name = Jerry, City Name =

student Name = Tom, City Name =

student Name = Senthil, City Name =

C# Program to Convert Big Endian to Little Endian

This C# Program Converts Big Endian to Little Endian. Here the big endian value is converted to little endian value.

Here is source code of the C# Program to Convert Big Endian to Little Endian. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Convert Big Endian to Little Endian*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
8. **using** System.Threading.Tasks;
10. **namespace** ConsoleApplication4
11. {
12. **class** Program
13. {
14. **static** **int** ReverseBytes(**int** val)
15. {
16. **byte**[] intAsBytes = BitConverter.GetBytes(val);
17. Array.Reverse(intAsBytes);
18. **return** BitConverter.ToInt32(intAsBytes, 0);
19. }
20. **static** **string** IntToBinaryString(**int** v)
21. {
22. **string** s = Convert.ToString(v, 2);
23. **string** t = s.PadLeft(32, '0');
24. **string** res = "";
25. **for** (**int** i = 0; i < t.Length; ++i)
26. {
27. **if** (i > 0 && i % 8 == 0)
28. res += " ";
29. res += t[i];
30. }
31. **return** res;
32. }
33. **static** **void** Main(**string**[] args)
34. {
35. **int** little = 2777;
36. **int** big = ReverseBytes(little);
37. **string** sLittle = IntToBinaryString(little);
38. **string** sBig = IntToBinaryString(big);
39. **int** oLittle = ReverseBytes(big);
40. **string** oString = IntToBinaryString(oLittle);
41. Console.WriteLine("Original (Intel) little endian value = "
42. + little);
43. Console.WriteLine("Original value as binary string = "
44. + sLittle);
45. Console.WriteLine("");
46. Console.WriteLine("Reversed big endian value = "
47. + big);
48. Console.WriteLine("Reversed value as string = "
49. + sBig);
50. Console.WriteLine("");
51. Console.ReadLine();
52. }
53. }
54. }

advertisements

Here is the output of the C# Program:

Original (Intel) little endian value = 2777

Original value as binary string = 00000000 00000000 00001010 11011001

Reversed big endian value = -653656064

Reversed value as string = 11011001 00001010 00000000 00000000

#### G. C# Programming Examples on Functions

This section contains C# programs on classes, abstract properties, method types, method hiding, namespaces and preprocessor attributes and various predefined functions.

#### 1. C# Examples on Predefined Functions

A group of statements which is used to perform a specific task is called a function. The built-in functions which perform standard operations are called pre-defined functions. The following C# Programs reverses and sorts a string using various predefined functions and the other programs use the concept of recursion to display the binary form of an integer and for calculating the sum of all the digits in a number.

C# Program to Reverse a String with Predefined Function

This C# Program Reverses a String with Predefined Function. Here the String is Reversed using the predefined Function reverse().

Here is source code of the C# Program to Reverse a String with Predefined Function. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Reverse a String with Predefined Function*
3. *\*/*
4. **using** System;
5. **class** linSearch
6. {
7. **public** **static** **void** Main()
8. {
9. Console.WriteLine("Enter Number of Elements you Want to Hold in the Array ? ");
10. **string** s = Console.ReadLine();
11. **int** x = Int32.Parse(s);
12. **int**[] a = new **int**[x];
13. Console.WriteLine("**\n** Enter Array Elements : ");
14. **for** (**int** i = 0; i < x; i++)
15. {
16. **string** s1 = Console.ReadLine();
17. a[i] = Int32.Parse(s1);
18. }
20. Array.Reverse(a);
21. Console.WriteLine("Reversed Array : ");
22. **for** (**int** i = 0; i < x; i++)
23. {
24. Console.WriteLine("Element {0} is {1}", i + 1, a[i]);
25. }
26. Console.Read();
27. }
28. }

advertisements

Here is the output of the C# Program:

Enter Number of Elements you Want to Hold in the Array ? 5

Enter Array Elements : 2

3

4

5

6

Reversed Array :

Element is : 6

Element is : 5

Element is : 4

Element is : 3

Element is : 2

C# Program to Sort a String using Predefined Function

This C# Program Sorts a String using Predefined Function. Here the string is sorted using the predefined function using sort().

Here is source code of the C# Program to Sort a String using Predefined Function. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Sort a String using Predefined Function*
3. *\*/*
4. **using** System;
5. **class** linSearch
6. {
7. **public** **static** **void** Main()
8. {
9. Console.WriteLine("Enter Number of Elements you Want to Hold in the Array ? ");
10. **string** s = Console.ReadLine();
11. **int** x = Int32.Parse(s);
12. **int**[] a = new **int**[x];
13. Console.WriteLine("Enter Array Elements :");
14. **for** (**int** i = 0; i < x; i++)
15. {
16. **string** s1 = Console.ReadLine();
17. a[i] = Int32.Parse(s1);
18. }
19. Array.Sort(a);
20. Console.WriteLine("Sorted Array : ");
21. **for** (**int** i = 0; i < x; i++)
22. {
23. Console.WriteLine("{0}",a[i]);
24. }
25. Console.Read();
26. }
27. }

advertisements

Here is the output of the C# Program:

Enter Number of Elements you Want to Hold in the Array ? 5

Enter Array Elements :

2

3

1

4

5

Sorted Array :

1

2

3

4

5

C# Program to Demonstrate Boxing Operations

This C# Program Demonstrates Boxing Operations. Here Boxing is the process of converting a value type to the type object or to any interface type implemented by this value type.

Here is source code of the C# Program to Demonstrate Boxing Operations. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Demonstrate Boxing Operations*
3. *\*/*
4. **using** System;
5. **class** sample
6. {
7. **int** x = 10;
8. **object** obj;
9. **void** boxmethod()
10. {
11. sample s= new sample();
12. **bool** b;
13. **object** ob="CSHARP";
14. b=s.obj is **int**;
15. Console.WriteLine(b);
16. s.obj = x;
17. b = s.obj is **int**;
18. Console.WriteLine("{0},{1},{2}",s.obj,s.x,b);
19. s.x = (**int**)s.obj;
20. s.x = 20;
21. b = s.obj is **int**;
22. Console.WriteLine("{0},{1},{2}", s.obj, s.x, b);
23. s.obj="CSHARP";
24. b=s.obj is **int**;
25. Console.WriteLine("{0},{1},{2}",s.obj,s.x,b);
26. Console.ReadLine();
27. }
28. **public** **static** **void** Main()
29. {
30. sample s=new sample();
31. s.boxmethod();
32. }
33. }

advertisements

Here is the output of the C# Program:

False

10,10,True

10,20,True

CSHARP,20,False

C# Program to Find Sum of Digits of a Number using Recursion

This C# Program Finds Sum of Digits of a Number using Recursion.

Here is source code of the C# Program to Find Sum of Digits of a Number using Recursion. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Find Sum of Digits of a Number using Recursion*
3. *\*/*
4. **using** System;
5. **class** program
6. {
7. **public** **static** **void** Main()
8. {
9. **int** num, result;
10. pro pg = new pro();
11. Console.WriteLine("Enter the Number : ");
12. num=**int**.Parse(Console.ReadLine());
13. result =pg.sum(num);
14. Console.WriteLine("Sum of Digits in {0} is {1}", num, result);
15. Console.ReadLine();
16. }
17. }
18. **class** pro
19. {
20. **public** **int** sum(**int** num)
21. {
22. **if** (num != 0)
23. {
24. **return** (num % 10 + sum(num / 10));
25. }
26. **else**
27. {
28. **return** 0;
29. }
30. }
31. }

advertisements

Here is the output of the C# Program:

Enter the Number :

234

Sum of Digits in 234 is 9

C# Program to Demonstrate the Operations of C# Path Class

This C# Program Demonstrates the Operations of C# Path Class. Here A path is a string that provides the location of a file or directory. A path does not necessarily point to a location on disk.

Here is source code of the C# Program to Demonstrate the Operations of C# Path Class. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Demonstrate the Operations of C# Path Class*
3. *\*/*
4. **using** System;
5. **using** System.IO;
6. **class** Test
7. {
8. **public** **static** **void** Main()
9. {
10. **string** p = @"c:\srip\sri.txt";
11. **string** p2 = @"c:\srip\sri";
12. **string** p3 = @"srip";
13. **if** (Path.HasExtension(p))
14. {
15. Console.WriteLine("{0} has an extension.", p);
16. }
17. **if** (!Path.HasExtension(p2))
18. {
19. Console.WriteLine("{0} has no extension.", p2);
20. }
21. **if** (!Path.IsPathRooted(p3))
22. {
23. Console.WriteLine("The string {0} contains no root information.", p3);
24. }
25. Console.WriteLine("Location for Temporary Files : {0}", Path.GetTempPath());
26. Console.WriteLine("Full path of {0} is {1}.", p3, Path.GetFullPath(p3));
27. Console.WriteLine("File available for Use : {0} ", Path.GetTempFileName());
28. Console.Read();
30. }
31. }

advertisements

Here is the output of the C# Program:

c:\srip\sri.txt has an extension.

c:\srip\sri has no extension.

The string srip contains no root information.

Location For Temporary Files : C:\Users\Win7\appdata\local\temp\

Full Path of Temp : D:\sri\ConsoleApplication22\ConsoleApplication22\bin\Debug\srip.

File Available for Use : C:\Users\Win7\appdata\local\temp\trmpEB1B.tmp

C# Program to Print Binary Equivalent of an Integer using Recursion

This C# Program Prints Binary Equivalent of an Integer using Recursion. Here This C# program, using recursion, finds the binary equivalent of a decimal number entered by the user. Decimal numbers are of base 10 while binary numbers are of base 2.

Here is source code of the C# Program to Print Binary Equivalent of an Integer using Recursion. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Print Binary Equivalent of an Integer using Recursion*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
8. **class** Program
9. {
10. **public** **static** **void** Main(**string**[] args)
11. {
12. **int** num;
13. prog pg = new prog();
14. Console.WriteLine("Enter a decimal number: ");
15. num = **int**.Parse(Console.ReadLine());
16. Console.WriteLine("The binary equivalent of num is :");
17. pg.binaryconversion(num);
18. Console.ReadLine();
19. }
20. }
21. **public** **class** prog
22. {
23. **public** **int** binaryconversion(**int** num)
24. {
25. **int** bin;
26. **if** (num != 0)
27. {
28. bin = (num % 2) + 10 \* binaryconversion(num / 2);
29. Console.Write(bin);
30. **return** 0;
31. }
32. **else**
33. {
34. **return** 0;
35. }
37. }
38. }

advertisements

Here is the output of the C# Program:

Enter a decimal number:

19

The binary equivalent of num is :

10011

#### 2. C# Examples on Classes and their Abstract Properties

A class is a blueprint from which individual objects are extracted. Abstract property does not contain any implementation of property accessors, it only includes the declaration that the class supports properties. iCollection count returns the number of elements in the collection. The following C# Programs demonstrates the properties of normal class and abstract classes, iCollection property and multiplication of numbers using recursion.

C# Program to find Product of 2 Numbers using Recursion

This C# Program finds Product of 2 Numbers using Recursion. Here the multiples of 3 and 5 are found and the sum of all the multiples are calculated and are displayed.

Here is source code of the C# Program to find Product of 2 Numbers using Recursion. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to find Product of 2 Numbers using Recursion*
3. *\*/*
4. **using** System;
5. **class** program
6. {
7. **public** **static** **void** Main()
8. {
9. **int** a, b, result;
10. Console.WriteLine("Enter two numbers to find their product: ");
11. a = **int**.Parse(Console.ReadLine());
12. b = **int**.Parse(Console.ReadLine());
13. prog pg = new prog();
14. result = pg.product(a, b);
15. Console.WriteLine("Product of {0} and {1} is {2}",a, b, result);
16. Console.ReadLine();
17. }
18. }
19. **class** prog
20. {
21. **public** **int** product(**int** a, **int** b)
22. {
23. **if** (a < b)
24. {
25. **return** product(b, a);
26. }
27. **else** **if** (b != 0)
28. {
29. **return** (a + product(a, b - 1));
30. }
31. **else**
32. {
33. **return** 0;
34. }
35. }
36. }

advertisements

Here is the output of the C# Program:

Enter two numbers to find their product:

5

6

Product of 5 and 6 is 30

C# Program to Produce a filtered sequence of elements that contain only one property of each Student

This C# Program to Produce a filtered sequence of elements that contain only one property of each Student using Select Clause LINQ. Here select clause specifies the type of values that will be produced when the query is executed.

Here is source code of the C# Program to Produce a filtered sequence of elements that contain only one property of each Student.The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Produce a filtered sequence of elements that contain only one property of each Student*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
8. **using** System.Threading.Tasks;
9. **class** Programs
10. {
11. **public** **class** Student
12. {
13. **public** **string** First { **get**; **set**; }
14. **public** **string** Last { **get**; **set**; }
15. **public** **int** ID { **get**; **set**; }
16. **public** List<**int**> Marks;
17. **public** ContactInfo GetContactInfo(Programs pg, **int** id)
18. {
19. ContactInfo allinfo =
20. (**from** ci **in** pg.contactList
21. **where** ci.ID == id
22. **select** ci)
23. .FirstOrDefault();
25. **return** allinfo;
26. }
28. **public** **override** **string** ToString()
29. {
30. **return** First + "" + Last + " : " + ID;
31. }
32. }
34. **public** **class** ContactInfo
35. {
36. **public** **int** ID { **get**; **set**; }
37. **public** **string** Email { **get**; **set**; }
38. **public** **string** Phone { **get**; **set**; }
39. **public** **override** **string** ToString() { **return** Email + "," + Phone; }
40. }
42. **public** **class** ScoreInfo
43. {
44. **public** **double** Average { **get**; **set**; }
45. **public** **int** ID { **get**; **set**; }
46. }
47. List<Student> students = new List<Student>()
48. {
49. new Student {First="Tom", Last=".S", ID=1, Marks= new List<**int**>() {97, 92, 81, 60}},
50. new Student {First="Jerry", Last=".M", ID=2, Marks= new List<**int**>() {75, 84, 91, 39}},
51. new Student {First="Bob", Last=".P", ID=3, Marks= new List<**int**>() {88, 94, 65, 91}},
52. new Student {First="Mark", Last=".G", ID=4, Marks= new List<**int**>() {97, 89, 85, 82}},
53. };
54. List<ContactInfo> contactList = new List<ContactInfo>()
55. {
56. new ContactInfo {ID=111, Email="Tom@abc.com", Phone="9328298765"},
57. new ContactInfo {ID=112, Email="Jerry123@aaa.com", Phone="9876543201"},
58. new ContactInfo {ID=113, Email="Bobstar@aaa.com", Phone="9087467653"},
59. new ContactInfo {ID=114, Email="Markantony@qqq.com", Phone="9870098761"}
60. };

63. **static** **void** Main(**string**[] args)
64. {
65. Programs pg = new Programs();
66. IEnumerable<**String**> studentQuery2 =
67. **from** student **in** pg.students
68. **where** student.ID > 1
69. **select** student.Last;
71. Console.WriteLine("**\r\n** studentQuery2: select range\_variable.Property");
72. **foreach** (**string** s **in** studentQuery2)
73. {
74. Console.WriteLine(s);
75. }
77. Console.ReadLine();
78. }
79. }

advertisements

Here is the output of the C# Program:

studentQuery2: select range\_variable.Property

.M

.P

.G

C# Program to Demonstrate Properties of the Class

This C# Program Demonstrates Properties of the Class. Here it demonstrates how properties are declared and used.

Here is source code of the C# Program to Demonstrate Properties of the Class. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Demonstrate Properties of the Class*
3. *\*/*
4. **using** System;
5. **class** Student
6. {
7. **private** **string** myName = "N/A";
8. **private** **int** myAge = 0;
9. **public** **string** Name
10. {
11. **get**
12. {
13. **return** myName;
14. }
15. **set**
16. {
17. myName = **value**;
18. }
19. }
20. **public** **int** Age
21. {
22. **get**
23. {
24. **return** myAge;
25. }
26. **set**
27. {
28. myAge = **value**;
29. }
30. }
32. **public** **override** **string** ToString()
33. {
34. **return** "Name = " + Name + ", Age = " + Age;
35. }
37. **public** **static** **void** Main()
38. {
39. Student Student = new Student();
40. Console.WriteLine("Student details - {0}", Student);
41. Student.Name = "BOB";
42. Student.Age = 99;
43. Console.WriteLine("Student details - {0}", Student);
44. Student.Age += 1;
45. Console.WriteLine("Student details - {0}", Student);
46. Console.ReadLine();
47. }
48. }

advertisements

Here is the output of the C# Program:

Student details - Name = N/A, Age = 0

Student details - Name = BOB, Age = 99

Student details - Name = BOB, Age = 100

C# Program to Demonstrate Abstract Properties

This C# Program Demonstrates Abstract Properties. Here An abstract property declaration does not provide an implementation of the property accessors.

Here is source code of the C# Program to Demonstrate Abstract Properties. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* CC# Program to Demonstrate Abstract Properties*
3. *\*/*
4. **using** System;
6. **public** **abstract** **class** Shape
7. {
8. **private** **string** myId;
10. **public** Shape(**string** s)
11. {
12. Id = s;
13. }
15. **public** **string** Id
16. {
17. **get**
18. {
19. **return** myId;
20. }
22. **set**
23. {
24. myId = **value**;
25. }
26. }
27. **public** **abstract** **double** Area
28. {
29. **get**;
30. }
32. **public** **override** **string** ToString()
33. {
34. **return** Id + " Area = " + **string**.Format("{0:F2}", Area);
35. }
36. }
37. **public** **class** Square : Shape
38. {
39. **private** **int** mySide;
41. **public** Square(**int** side, **string** id)
42. : **base**(id)
43. {
44. mySide = side;
45. }
47. **public** **override** **double** Area
48. {
49. **get**
50. {
51. *// Given the side, return the area of a square:*
52. **return** mySide \* mySide;
53. }
54. }
55. }
57. **public** **class** Circle : Shape
58. {
59. **private** **int** myRadius;
61. **public** Circle(**int** radius, **string** id)
62. : **base**(id)
63. {
64. myRadius = radius;
65. }
67. **public** **override** **double** Area
68. {
69. **get**
70. {
71. *// Given the radius, return the area of a circle:*
72. **return** myRadius \* myRadius \* System.Math.PI;
73. }
74. }
75. }
77. **public** **class** Rectangle : Shape
78. {
79. **private** **int** myWidth;
80. **private** **int** myHeight;
82. **public** Rectangle(**int** width, **int** height, **string** id)
83. : **base**(id)
84. {
85. myWidth = width;
86. myHeight = height;
87. }
89. **public** **override** **double** Area
90. {
91. **get**
92. {
93. *// Given the width and height, return the area of a rectangle:*
94. **return** myWidth \* myHeight;
95. }
96. }
97. }
98. **public** **class** TestClass
99. {
100. **public** **static** **void** Main()
101. {
102. Shape[] shapes =
103. {
104. new Square(5, "Square #1"),
105. new Circle(3, "Circle #1"),
106. new Rectangle( 4, 5, "Rectangle #1")
107. };
109. System.Console.WriteLine("Shapes Collection");
110. **foreach** (Shape s **in** shapes)
111. {
112. System.Console.WriteLine(s);
113. }
114. Console.ReadLine();
115. }
116. }

advertisements

Here is the output of the C# Program:

Shapes Collection

Square #1 Area = 25.00

Circle #1 Area = 28.27

Rectangle #1 Area = 20.00

C# Program to Demonstrate Environment Type

This C# Program Demonstrates Environment Type. Here Environment.Exit does not run finally statements. It provides a way to kill a process in its physical sense on the operating system..

Here is source code of the C# Program to Demonstrate Environment Type. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Demonstrate Environment Type*
3. *using System;*
4. *class Program*
5. *{*
6. *static void Main()*
7. *{*
8. *try*
9. *{*
10. *Environment.Exit(0);*
11. *}*
12. *finally*
13. *{*
14. *Console.WriteLine("Statement that is Never Reached");*
15. *}*
16. *Console.Read();*
17. *}*
18. *}*

advertisements

Here is the output of the C# Program:

// No Output is obtained

// Calls the Environment.Exit method and returns a zero status code.

// The finally statement is never reached.

C# Program to Demonstrate the icollection Count

This C# Program Demonstrates the icollection Count. Here the count is found.

Here is source code of the C# Program to Demonstrate the icollection Count. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Demonstrate the icollection Count*
3. *\*/*
4. **using** System;
5. **using** System.Collections;
7. **public** **class** CountArray
8. {
9. **public** **static** **void** Main()
10. {
11. **string**[] strings = { "Ajax", "Atlas" ,"a","b"};
12. DisplayCollectionProperty(strings);
13. Console.ReadLine();
14. }
16. **public** **static** **void** DisplayCollectionProperty
17. (ICollection iCollection)
18. {
19. Console.WriteLine("Count = {0}", iCollection.Count);
20. }
21. }

advertisements

Here is the output of the C# Program:

Count=4

C# Program to Demonstrate iscollection.synchronised

This C# Program Demonstrates iscollection.synchronised. Here the synchronisation operation is demonstrated.

Here is source code of the C# Program to Demonstrate iscollection.synchronised. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Demonstrate iscollection.synchronised*
3. *\*/*
4. **using** System;
5. **using** System.Collections;
7. **public** **class** CountArray
8. {
9. **public** **static** **void** Main()
10. {
11. **string**[] strings = { "Ajax", "Atlas" };
12. DisplayCollectionProperty(strings);
13. Console.ReadLine();
14. }
16. **public** **static** **void** DisplayCollectionProperty
17. (ICollection iCollection)
18. {
19. Console.WriteLine("IsSynchronized: {0}",
20. iCollection.IsSynchronized);
21. }
22. }

advertisements

Here is the output of the C# Program:

IsSynchronized: False

#### 3. C# Examples on Method Types and Method Hiding

Method hiding is a phenomenon in which a subclass has defined a class method with the same signature as a class method in the superclass. In that case the superclass method is hidden by the subclass. Goto statement is used for modifying the normal sequence of program execution by transferring control to some other part of the program. Static method can be invoked without creating an instance of a class. The method without any name is called anonymous method. The Obsolete attribute marks a program entity as one that is no longer recommended for use. The CultureInfo class contains culture-specific information, such as the associated language, sub-language, country/region, calendar, and cultural conventions. The following C# programs demonstrate the various properties mentioned above.

C# Program Illustrate Method Hiding

This C# Program Illustrates Method Hiding. Here a method func() in the base class and in the derived class give a completely new definition to the same method by preceding it with ‘new’ keyword

Here is source code of the C# Program Illustrate Method Hiding. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program Illustrate Method Hiding*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Text;
7. **namespace** ConsoleApplication1
8. {
9. **public** **class** Demo
10. {
11. **public** **virtual** **double** Area(**double** r)
12. {
13. **return** r \* r;
14. }
15. **public** **void** func()
16. {
17. Console.WriteLine("Base Class");
18. }
19. }
20. **public** **class** A : Demo
21. {
22. **public** **override** **double** Area(**double** r)
23. {
25. **return** **base**.Area(r) \* r;
26. }
27. **public** new **void** func()
28. {
29. Console.WriteLine("Derived Class");
30. }
31. }
32. **public** **class** Test
33. {
34. **public** **static** **void** Main(**string**[] args)
35. {
36. A o1 = new A();
37. Console.WriteLine(o1.Area(20));
38. o1.func();
39. Console.ReadLine();
40. }
41. }
42. }

advertisements

Here is the output of the C# Program:

8000

Derived Class

C# Program to Illustrate the Concept of Goto

This C# Program Illustrates the Concept of Goto. Here The goto statement transfers the program control directly to a labeled statement that is Loop.

Here is source code of the C# Program to Illustrate the Concept of Goto. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Illustrate the Concept of Goto*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
8. **namespace** Example
9. {
10. **class** Program
11. {
12. **static** **void** Main(**string**[] args)
13. {
14. **int** no, per, option;
15. **float** ans;
16. loop:
17. Console.Write("Enter a Number :**\t**");
18. no = Convert.ToInt32(Console.ReadLine());
19. Console.Write("**\n**Enter Percentage Value : **\t**");
20. per = Convert.ToInt32(Console.ReadLine());
21. ans = (**float**)(no \* per) / 100;
22. Console.WriteLine("Percentage Value is:**\t**{0}", ans);
23. Console.Write("**\n**Calculate again press 1. To quit press digit:**\t**");
24. option = Convert.ToInt32(Console.ReadLine());
25. **if** (option == 1)
26. {
27. **goto** loop;
28. }
29. Console.WriteLine("Press Enter for quit");
30. Console.ReadLine();
31. }
32. }
33. }

advertisements

Here is the output of the C# Program:

Enter a number : 320

Enter Percentage Value : 10

Percentage value is : 32

Calculate again press 1. To quit Press digit: 1

Enter a number : 730

Enter Percentage Value: 10

Percentage value is: 73

Calculate again press 1. To quit press digit: 6

Press Enter for quit

C# Program to Create Anonymous Method

This C# Program Creates Anonymous Method. Here By this anonymou methods,reduce the coding overhead in instantiating delegates because a separate method need not be created.

Here is source code of the C# Program to Create Anonymous Method. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Create Anonymous Method*
3. *\*/*
4. **using** System;
5. **delegate** **void** Print(**string** s);
6. **class** TestClass
7. {
8. **static** **void** Main()
9. {
10. Print obj = **delegate**(**string** j)
11. {
12. System.Console.WriteLine(j);
13. };
14. obj("Delegate Using the Anonymous Method");
15. obj = new Print(TestClass.named);
16. obj("Delegate Using the Named Method");
17. Console.Read();
18. }
19. **static** **void** named(**string** k)
20. {
21. System.Console.WriteLine(k);
22. }
24. }

advertisements

Here is the output of the C# Program:

Delegate Using the Anonymous Method

Delegate Using the Named Method

C# Program to Implement Static Method

This C# Program Implements Static Method. Here Static methods have no instances. They are called with the type name, not an instance identifier.

Here is source code of the C# Program to Implement Static Method. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Implement Static Method*
3. *\*/*
4. **using** System;
5. **class** Program
6. {
7. **static** **void** stamethod()
8. {
9. Console.WriteLine("Static Method");
10. }
11. **void** MethodB()
12. {
13. Console.WriteLine("Instance Method");
14. }
15. **static** **char** stamethod2()
16. {
17. Console.WriteLine("Static Method");
18. **return** 'C';
19. }
20. **static** **void** Main()
21. {
22. Program.stamethod();
23. Console.WriteLine(Program.stamethod2());
24. Program programInstance = new Program();
25. programInstance.MethodB();
26. Console.Read();
27. }
28. }

advertisements

Here is the output of the C# Program:

Static Method

Static Method

C

Instance Method

C# Program to Demonstrate Culture Names

This C# Program Demonstrates Culture Names. Here The CultureInfo object that is returned by this property and its associated objects determine the default format for dates, times, numbers, currency values, the sorting order of text, casing conventions, and string comparisons.

Here is source code of the C# Program to Demonstrate Culture Names. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Demonstrate Culture Names*
3. *\*/*
4. **using** System;
5. **using** System.Globalization;
6. **using** System.Threading;
7. **public** **class** Info : MarshalByRefObject
8. {
9. **public** **void** ShowCurrentCulture()
10. {
11. Console.WriteLine("Culture of {0} in application domain {1}: {2}",Thread.CurrentThread.Name,AppDomain.CurrentDomain.FriendlyName,CultureInfo.CurrentCulture.Name);
12. }
13. }
14. **public** **class** Example
15. {
16. **public** **static** **void** Main()
17. {
18. Info inf = new Info();
19. Thread.CurrentThread.Name = "MainThread";
20. Thread.CurrentThread.CurrentCulture = CultureInfo.CreateSpecificCulture("nl-NL");
21. inf.ShowCurrentCulture();
22. AppDomain ad = AppDomain.CreateDomain("Domain2");
23. Info inf2 = (Info)ad.CreateInstanceAndUnwrap(typeof(Info).Assembly.FullName, "Info");
24. inf2.ShowCurrentCulture();
25. Console.ReadLine();
26. }
27. }

advertisements

Here is the output of the C# Program:

Culture of MainThread in application domain ConsoleApplication32.vshoot.exe : nl-NL

Culture of MainThread in application domain Domain2 : nl-NL

C# Program to Create Obsolete Class

This C# Program Creates Obsolete Class. Here The Obsolete attribute generates a compile-time warning. When a method has the Obsolete attribute, the C# compiler issues a warning if it is called.

Here is source code of the C# Program to Create Obsolete Class. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Create Obsolete Class*
3. *\*/*
4. **using** System;
5. **class** Program
6. {
7. **static** **void** Main()
8. {
9. MethodA();
10. MethodB();
11. Console.Read();
12. }
13. [Obsolete("Use MethodB Instead")]
14. **static** **void** MethodA()
15. {
16. }
17. **static** **void** MethodB()
18. {
19. Console.WriteLine(" MethodA shows an Warning when called and MethodB is not an Obsolete Method ");
20. }
21. }

advertisements

Here is the output of the C# Program:

MethodA shows an Warning when called and MethodB is not an Obsolete Method

C# Program to Illustrate the use of Conditional Logical Operator

This C# Program Illustrates the use of Conditional Logical Operators. Here the the ternary operator (?:) is a conditional operator. It is a convenient operator for cases where one of two values are to be picked, depending on the conditional expression.

Here is source code of the C# Program to Illustrate the use of Conditional Logical Operators. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Illustrate the use of Conditional Logical Operator*
3. *\*/*
4. **using** System;
5. **public** **class** Program
6. {
7. **static** **void** Main()
8. {
9. **int** age;
10. Console.WriteLine("Enter the Age :");
11. age=**int**.Parse(Console.ReadLine());
12. **bool** adult = age >= 18 ? **true** : **false**;
13. Console.WriteLine("Adult : {0}", adult);
14. Console.Read();
15. }
16. }

advertisements

Here is the output of the C# Program:

Enter the Age : 10

Adult : false

#### 4. C# Examples on Namespaces and Preprocessor Attributes

Nullable data type assigns null values to data types. #define preprocessor substitutes a preprocessor macro. A namespace is a collection of pages which have content with a similar purpose. The following C# Programs demonstrate pass by value, pass by reference, namespaces, preprocessors new modifier and various nullable data types.

C# Program to Demonstrate Pass by Value Parameter

This C# Program Demonstrates Pass by Value Parameter . Here Passing a value-type variable to a method by value means passing a copy of the variable to the method. Any changes to the parameter that take place inside the method have no affect on the original data stored in the argument variable.

Here is source code of the C# Program to Demonstrate Pass by Value Parameter . The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Demonstrate Pass by Value Parameter*
3. *\*/*
4. **using** System;
5. **class** program
6. {
7. **static** **void** Cube(**int** x)
8. {
9. x = x \* x \* x;
10. Console.WriteLine("Value Within the Cube method : {0}", x);
11. }
12. **static** **void** Main()
13. {
14. **int** num = 5;
15. Console.WriteLine("Value Before the Method is called : {0}", num);
16. Cube(num);
17. Console.WriteLine("Value After the Method is called : {0}", num);
18. Console.ReadKey();
19. }
20. }

advertisements

Here is the output of the C# Program:

Value Before the Method is called : 5

Value Within the Cube method : 125

Value After the Method is called : 5

C# Program to Print Hello World Without using WriteLine

This C# Program Prints Hello World Without using WriteLine. Here the text is printed without using any writeline functions.

Here is source code of the C# Program to Print Hello World Without using WriteLine. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Print Hello World Without using WriteLine*
3. *\*/*
4. **using** System;
5. **class** Program
6. {
7. **static** **void** Main(**string**[] args)
8. {
9. **if** (System.Console.OpenStandardOutput().BeginWrite(new **byte**[] { 072, 101, 108, 108, 111, 032, 087, 111, 114, 108, 100, 0 },0, 12, **null**, **null**).AsyncWaitHandle.WaitOne())
10. {
11. }
12. **if** (System.Console.ReadKey().Modifiers == 0)
13. {
14. }
15. }
16. }

advertisements

Here is the output of the C# Program:

Hello World

C# Program to Demonstrate Pass by Reference Parameter

This C# Program Demonstrates Pass by Reference Parameter . Here Passing by reference enables function members, methods to change the value of the parameters and have that change persist in the calling environment. To pass a parameter by reference, use the ref or out keyword.

Here is source code of the C# Program to Demonstrate Pass by Reference Parameter . The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Demonstrate Pass by Reference Parameter*
3. *\*/*
4. **using** System;
5. **class** Program
6. {
7. **static** **void** Main(**string**[] args)
8. {
9. **int** val;
10. val = 4;
11. Console.WriteLine("Value Before : {0}", val);
12. square(**ref** val);
13. Console.WriteLine("Value After : {0}", val);
14. Console.Read();
15. }
16. **static** **void** square(**ref** **int** refParam)
17. {
18. refParam \*= refParam;
19. }
20. }

advertisements

Here is the output of the C# Program:

Value Before : 4

Value After : 16

C# Program to Implement Namespaces

This C# Program Implements Namespaces. Here namespaces are used to logically arrange classes, structs, interfaces, enums and delegates and keyword namespace is required to create a user defined name space.

Here is source code of the C# Program to Implement Namespaces. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Implement Namespaces*
3. *\*/*
4. **using** System;
5. **namespace** Sanfoundry.Csharp.Codes
6. {
7. **class** TestClass
8. {
9. **public** TestClass()
10. {
11. Console.WriteLine("Class to Demonstrate Namespace");
12. }
13. }
14. }
15. **class** MyClient
16. {
17. **public** **static** **void** Main()
18. {
19. Sanfoundry.Csharp.Codes.TestClass mc = new Sanfoundry.Csharp.Codes.TestClass();
20. Console.ReadLine();
21. }
22. }

advertisements

Here is the output of the C# Program:

Class to Demonstrate Namespace

C# Program to Demonstrate the Working #define Preprocessor

This C# Program Demonstrates the Working #define Preprocessor. Here The text defines the symbol B, then defines the symbol A, then undefines the symbol A. The program will compile with B being defined, and A being undefined.

Here is source code of the C# Program to Demonstrate the Working #define Preprocessor. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Demonstrate the Working #define Preprocessor*
3. *\*/*
4. #define B
5. #define A
6. #undef A
7. **using** System;
8. **class** Program
9. {
10. **static** **void** Main()
11. {
13. #if A
14. Console.WriteLine("'A' is Displayed Based on the undef Directive ");
15. #elif B
16. Console.WriteLine("'B' is Displayed Based on the undef Directive");
17. #endif
18. Console.ReadLine();
19. }
20. }

advertisements

Here is the output of the C# Program:

'B' is Displayed Based on the undef Directive

C# Program to Illustrate Regular Expression Pattern

This C# Program Illustrates Regular Expression Pattern. Here the The Regex class is used for representing a regular expression and the words that start with the given letter is displayed.

Here is source code of the C# Program to Illustrate Regular Expression Pattern. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Illustrate Regular Expression Pattern*
3. *\*/*
4. **using** System;
5. **using** System.Text.RegularExpressions;
6. **namespace** Application
7. {
8. **class** Program
9. {
10. **private** **static** **void** showMatch(**string** text, **string** expr)
11. {
12. Console.WriteLine("The Expression : " + expr);
13. MatchCollection m = Regex.Matches(text, expr);
14. **foreach** (Match m1 **in** m)
15. {
16. Console.WriteLine(m1);
17. }
18. }
19. **static** **void** Main(**string**[] args)
20. {
21. **string** str = "Sanfoundry , a high end Technology Training company";
22. Console.WriteLine("Matching words that start with 'S': ");
23. showMatch(str, @"\bS\S\*");
24. Console.ReadKey();
25. }
26. }
27. }

advertisements

Here is the output of the C# Program:

Matching Words that Starts With 'S' :

The Expression : \bs\s\*

Sanfoundry

C# Program to Ilustrate Nullable Data Types

This C# Program to Ilustrate Nullable Data Types. Here C# provides a special data types, the nullable types, to which you can assign normal range of values as well as null values.

Here is source code of the C# Program to Ilustrate Nullable Data Types. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Ilustrate Nullable Data Types*
3. *\*/*
4. **using** System;
5. **namespace** Application
6. {
7. **class** Nullables
8. {
9. **static** **void** Main(**string**[] args)
10. {
11. **int**? num1 = **null**;
12. **int**? num2 = 100;
13. **double**? num3 = new **double**?();
14. **double**? num4 = 3.14157;
15. **bool**? boolval = new **bool**?();
16. Console.WriteLine("Nullables : {0}, {1}, {2}, {3}",
17. num1, num2, num3, num4);
18. Console.WriteLine("A Nullable boolean value: {0}", boolval);
19. Console.ReadLine();
21. }
22. }
23. }

advertisements

Here is the output of the C# Program:

Nullables : ,100,,3.1457

A Nullable Boolean Value :

C# Program to Demonstrate New Modifier

This C# Program Demonstrates New modifier. Here the new modifier specifies that a method is supposed to hide a base method. It eliminates a warning issued by the compiler.

Here is source code of the C# Program to Demonstrate New modifier. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Demonstrate New modifier*
3. *\*/*
4. **using** System;
6. **class** TEST
7. {
8. **public** **void** func()
9. {
10. Console.WriteLine("TEST.func");
11. }
12. }
14. **class** TEST1 : TEST
15. {
16. **public** new **void** func()
17. {
18. Console.WriteLine("TEST1.func");
19. }
20. }
22. **class** Program
23. {
24. **static** **void** Main()
25. {
26. TEST ref1 = new TEST();
27. TEST ref2 = new TEST1();
28. TEST1 ref3 = new TEST1();
30. ref1.func();
31. ref2.func();
32. ref3.func();
33. Console.Read();
34. }
35. }

advertisements

Here is the output of the C# Program:

TEST.func

TEST.func

TEST1.func

#### H. C# Programming Examples on Delegates

The C# programs in this section illustrate the usage of delegates, types of delegates and the basic functions of delegates.

#### 1. C# Examples illustrating the Basic Functions of Delegates

Delegate in C# is similar to pointer in C. It is a reference type variable which contains a reference to a method. The following programs combine 2 delegates, demonstrates an array of delegates and displays the corresponding results using delegates.

C# Program to Combine Two Delegates

This C# Program to Combine Two Delegates. Here When the multicast delegate is called, it invokes the delegates in the list, in order. Only delegates of the same type can be combined.

Here is source code of the C# Program to Combine Two Delegates. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Combine Two Delegates*
3. *\*/*
4. **using** System;
5. **delegate** **void** dele(**string** s);
6. **class** TestClass
7. {
8. **static** **void** Good(**string** s)
9. {
10. System.Console.WriteLine(" Good, {0}!", s);
11. }
13. **static** **void** Morning(**string** s)
14. {
15. System.Console.WriteLine(" Morning, {0}!", s);
16. }
18. **static** **void** Main()
19. {
20. dele firstdel, secondDel, multiDel, multiMinusfirstdel;
21. firstdel = Good;
22. secondDel = Morning;
23. multiDel = firstdel + secondDel;
24. multiMinusfirstdel = multiDel - firstdel;
25. Console.WriteLine("Invoking delegate firstdel:");
26. firstdel("A");
27. Console.WriteLine("Invoking delegate secondDel:");
28. secondDel("B");
29. Console.WriteLine("Invoking delegate multiDel:");
30. multiDel("C");
31. Console.WriteLine("Invoking delegate multiMinusfirstdel:");
32. multiMinusfirstdel("D");
33. Console.ReadLine();
34. }
35. }

advertisements

Here is the output of the C# Program:

Invoking delegate firstDel:

Good, A!

Invoking delegate SecondDel:

Morning, B!

Invoking delegate multiDel:

Good, C!

Morning, C!

Invoking delegate multiMinusFirstDel:

Morning, D!

C# Program to Illustrate Array of Delegates

This C# Program Illustrates Array of Delegates. Here an array of delegate is created similar to that of normal declaration of the delegate.

Here is source code of the C# Program to Illustrate Array of Delegates. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Illustrate Array of Delegates*
3. *\*/*
4. **using** System;
5. **delegate** **double** Measure(**double** R);
6. **public** **class** Circle
7. {
8. **const** **double** PI = 3.14159;
9. **public** **double** Diameter(**double** Radius)
10. {
11. **return** Radius \* 2;
12. }
14. **public** **double** Circumference(**double** Radius)
15. {
16. **return** Diameter(Radius) \* PI;
17. }
19. **public** **double** Area(**double** Radius)
20. {
21. **return** Radius \* Radius \* PI;
22. }
23. }
24. **public** **static** **class** Program
25. {
26. **static** **int** Main()
27. {
28. **double** R = 10;
29. Circle circ = new Circle();
30. Measure[] Calc = new Measure[3];
31. Calc[0] = new Measure(circ.Diameter);
32. **double** D = Calc[0](R);
33. Calc[1] = new Measure(circ.Circumference);
34. **double** C = Calc[1](R);
35. Calc[2] = new Measure(circ.Area);
36. **double** A = Calc[2](R);
37. Console.WriteLine("Diameter: {0}", D);
38. Console.WriteLine("Circumference: {0}", C);
39. Console.WriteLine("Area: {0}**\n**", A);
40. Console.ReadLine();
41. **return** 0;
42. }
43. }

advertisements

Here is the output of the C# Program:

Diameter : 20

Circumference : 62.8318

Area : 314.159

C# Program to Display Results using Delegates

This C# Program Displays Results using Delegates. Here Delegate is a type which holds the method(s) reference in an object. It is also referred to as a type safe function pointer.

Here is source code of the C# Program to Display Results using Delegates. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Display Results using Delegates*
3. *\*/*
4. **using** System;
5. **public** **class** example
6. {
7. **public** **delegate** **int** DelegateHandler(**int** a, **int** b);
8. **static** **void** Main(**string**[] args)
9. {
10. Results Results = new Results();
11. DelegateHandler sum = new DelegateHandler(Results.sum);
12. **int** result = sum(50, 20);
13. Console.WriteLine("Result is: " + result);
14. Console.ReadLine();
15. }
16. }
18. **public** **class** Results
19. {
20. **public** **int** sum(**int** a, **int** b)
21. {
22. **return** a + b;
23. }
24. }

advertisements

Here is the output of the C# Program:

Result is: 70

#### 2. C# Examples illustrating the types of Delegates

A delegate can define its own type parameters. The generic delegate can specify the type argument to create a closed constructed type. The multicast delegate consists of a list of the assigned delegates. When the multicast delegate is called, it invokes the other delegates in the list in order. Only delegates of the same type can be combined. The following programs creates generic delegates, implements delegates and multicast delegates and declares and instantiates delegates.

C# Program to Create Generic Delegate

This C# Program Creates Generic Delegate. Here Code that references the generic delegate can specify the type argument to create a closed constructed type, just like when instantiating a generic class or calling a generic method.

Here is source code of the C# Program to Create Generic Delegate. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Create Generic Delegate*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **delegate** T NumberChanger<T>(T n);
7. **namespace** GenericDelegateAppl
8. {
9. **class** TestDelegate
10. {
11. **static** **int** num = 10;
12. **public** **static** **int** AddNum(**int** p)
13. {
14. num += p;
15. **return** num;
16. }
18. **public** **static** **int** MultNum(**int** q)
19. {
20. num \*= q;
21. **return** num;
22. }
23. **public** **static** **int** getNum()
24. {
25. **return** num;
26. }
28. **static** **void** Main(**string**[] args)
29. {
30. NumberChanger<**int**> nc1 = new NumberChanger<**int**>(AddNum);
31. NumberChanger<**int**> nc2 = new NumberChanger<**int**>(MultNum);
32. nc1(25);
33. Console.WriteLine("Value of Num: {0}", getNum());
34. nc2(5);
35. Console.WriteLine("Value of Num: {0}", getNum());
36. Console.ReadKey();
37. }
38. }
39. }

advertisements

Here is the output of the C# Program:

Result of the Addition : 35

Result of the Product : 350

C# Program to Implement Multicast Delegates

This C# Program Implements Multicast Delegates. Here Multicast delegate is a delegate which holds a reference to more than one method.

Here is source code of the C# Program to Implement Multicast Delegates. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Implement Multicast Delegates*
3. *\*/*
4. **using** System;
5. **delegate** **void** dele(**int** a, **int** b);
6. **public** **class** Oper
7. {
8. **public** **static** **void** **Add**(**int** a, **int** b)
9. {
10. Console.WriteLine("{0} + {1} = {2}", a, b, a + b);
11. }
13. **public** **static** **void** Sub(**int** a, **int** b)
14. {
15. Console.WriteLine("{0} - {1} = {2}", a, b, a - b);
16. }
17. }
18. **public** **class** program
19. {
20. **static** **void** Main()
21. {
22. dele del = new dele(Oper.**Add**);
23. del += new dele(Oper.Sub);
24. del(4, 2);
25. del -= new dele(Oper.Sub);
26. del(1, 9);
27. Console.Read();
28. }
29. }

advertisements

Here is the output of the C# Program:

4 + 2 = 6

4 - 2 = 2

1 + 9 = 10

C# Program to Implement Delegates

This C# Program Implements Delegates. Here the delegate is created first and some data is displayed using the delegate.

Here is source code of the C# Program to Implement Delegates. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Implement Delegates*
3. *\*/*
4. **using** System;
5. **using** System.IO;
6. **delegate** **void** delegatewriter(**string** message);
7. **class** delgwriter
8. {
9. StreamWriter w;
10. **public** delgwriter(**string** path)
11. {
12. w = File.CreateText(path);
13. }
14. **public** **void** display(**string** msg)
15. {
16. w.WriteLine(msg);
17. }
18. **public** **void** Flush()
19. {
20. w.Flush();
21. }
22. **public** **void** Close()
23. {
24. w.Close();
25. }
26. }
27. **class** Test
28. {
29. **static** delegatewriter delgwri;
30. **static** **void** display(**string** s)
31. {
32. Console.WriteLine(s);
33. }
34. **static** **void** Main(**string**[] arg)
35. {
36. delgwriter x = new delgwriter("log.txt");
37. delgwri += new delegatewriter(display);
38. delgwri += new delegatewriter(x.display);
39. delgwri("C");
40. delgwri("C++");
41. delgwri("Java");
42. x.Flush();
43. x.Close();
44. Console.Read();
45. }
46. }

advertisements

Here is the output of the C# Program:

C

C++

Java

C# Program to Declare and Instantiate Delegates

This C# Program to Declare and Instantiate Delegates. Here Delegates are often used to implement callbacks and event listeners.

Here is source code of the C# Program to Declare and Instantiate Delegates. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Declare and Instantiate Delegates*
3. *\*/*
4. **using** System;
5. **delegate** **void** dele1();
6. **public** **class** Delegateintro
7. {
8. **static** **void** Main()
9. {
10. dele1 del = new dele1(Write);
11. del();
12. }
13. **static** **void** Write()
14. {
15. Console.WriteLine("Calling Write ");
16. }
17. }

advertisements

Here is the output of the C# Program:

Calling Write

#### 3. C# Examples illustrating the use of Delegates in Mathematics

The programs in this section perform math operations using delegates, converts feet to inches using delegates. The other programs in the section implements principles of delegates, implements arithmetic operations like Addition, Subtraction, Multiplication and Division using delegates.

C# Program to Call Math Operations using Delegates

This C# Program Calls Math Operations using Delegates. Here the Delegates are used to encapsulate functions into callable function objects that are used much as function pointers are used in C, C++, and other languages. Like function pointers, delegates enable you to separate a function reference from its implementation, allowing the implementation to exist in a separate module.

Here is source code of the C# Program to Call Math Operations using Delegates. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Call Math Operations using Delegates*
3. *\*/*
5. **using** System;
6. **public** **class** MathOperations
7. {
8. **public** **static** **double** Multiply(**double** **value**)
9. {
10. **return** **value** \* 2;
11. }
13. **public** **static** **double** Square(**double** **value**)
14. {
15. **return** **value** \* **value**;
16. }
17. }

20. **delegate** **double** DoubleOp(**double** x);
22. **class** Application
23. {
24. **static** **void** Main()
25. {
26. DoubleOp[] operations =
27. {
28. MathOperations.Multiply,
29. MathOperations.Square
30. };
32. **for** (**int** i = 0; i < operations.Length; i++)
33. {
34. Console.WriteLine("Operation[{0}]:", i);
35. ProcessAndDisplayNumber(operations[i], 5.0);
36. ProcessAndDisplayNumber(operations[i], 13.55);
37. ProcessAndDisplayNumber(operations[i], 1.732);
38. Console.WriteLine();
39. }
40. Console.ReadLine();
41. }
43. **static** **void** ProcessAndDisplayNumber(DoubleOp action, **double** **value**)
44. {
45. **double** result = action(**value**);
46. Console.WriteLine(
47. "Value : {0} Result : {1}", **value**, result);
48. }
49. }

advertisements

Here is the output of the C# Program:

Operation[0]:

Value : 5 Result : 10

Value : 13.55 Result : 27.1

Value : 1.732 Result : 3.464

Operation[1]:

Value : 5 Result : 25

Value : 13.55 Result : 183.6025

Value : 1.732 Result : 2.999824

C# Program to Convert Feet to Inches using Delegates

This C# Program Converts Feet to Inches using Delegates. Here Delegates are used to pass methods as arguments to other methods. After passing the arguments the values are converted to the required form

Here is source code of the C# Program to Convert Feet to Inches using Delegates. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Convert Feet to Inches using Delegates*
3. *\*/*
4. **using** System;
5. **public** **delegate** **double** Conversion(**double** **from**);
6. **class** DelegateDemo
7. {
8. **public** **static** **double** FeetToInches(**double** feet)
9. {
10. **return** feet \* 12;
11. }
13. **static** **void** Main()
14. {
16. Conversion doConversion = new Conversion(FeetToInches);
17. Console.Write("Enter Feet: ");
18. **double** feet = **Double**.Parse(Console.ReadLine());
19. **double** inches = doConversion(feet);
20. Console.WriteLine("**\n**{0} Feet = {1} Inches.**\n**", feet, inches);
21. Console.ReadLine();
22. }
23. }

advertisements

Here is the output of the C# Program:

Enter Feet: 50

50 Feet = 600 Inches.

C# Program to Implement Principles of Delegates

This C# Program Implements Principles of Delegates. Here delegate is a type that represents references to methods with a particular parameter list and return type. When you instantiate a delegate, you can associate its instance with any method with a compatible signature and return type. You can invoke (or call) the method through the delegate instance.  
Here is source code of the C# Program to Implement Principles of Delegates. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Implement Principles of Delegates*
3. *\*/*
5. **using** System;
6. **class** Program
7. {
8. **delegate** **string** UppercaseDelegate(**string** input);
9. **static** **string** UppercaseFirst(**string** input)
10. {
11. **char**[] buffer = input.ToCharArray();
12. buffer[0] = **char**.ToUpper(buffer[0]);
13. **return** new **string**(buffer);
14. }
15. **static** **string** UppercaseLast(**string** input)
16. {
17. **char**[] buffer = input.ToCharArray();
18. buffer[buffer.Length - 1] = **char**.ToUpper(buffer[buffer.Length - 1]);
19. **return** new **string**(buffer);
20. }
21. **static** **string** UppercaseAll(**string** input)
22. {
23. **return** input.ToUpper();
24. }
25. **static** **void** WriteOutput(**string** input, UppercaseDelegate del)
26. {
27. Console.WriteLine("Input String: {0}", input);
28. Console.WriteLine("Output String: {0}", del(input));
29. }
30. **static** **void** Main()
31. {
32. WriteOutput("tom ", new UppercaseDelegate(UppercaseFirst));
33. WriteOutput("tom", new UppercaseDelegate(UppercaseLast));
34. WriteOutput("tom", new UppercaseDelegate(UppercaseAll));
35. Console.ReadLine();
36. }
37. }

Here is the output of the C# Program:

advertisements

20369925375650a2ab19539\_000001

C# Program to Implement Arithmetic Operations using Delegates

This C# Program Implements Arithmetic Operations using Delegates. Here the delegate is a form of type-safe function pointer used by the .NET Framework. Delegates are often used to implement callbacks and event listeners. A delegate does not need to know anything about classes of methods it works with.

Here is source code of the C# Program to Implement Arithmetic Operations using Delegates. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Implement Arithmetic Operations using Delegates*
3. *\*/*
4. **using** System;
5. **delegate** **int** NumberChanger(**int** n);
6. **namespace** example
7. {
8. **class** **Delegate**
9. {
10. **static** **int** num = 10;
11. **public** **static** **int** AddNum(**int** a)
12. {
13. num += a;
14. **return** num;
15. }
17. **public** **static** **int** MultNum(**int** b)
18. {
19. num \*= b;
20. **return** num;
21. }
22. **public** **static** **int** getNum()
23. {
24. **return** num;
25. }
27. **static** **void** Main(**string**[] args)
28. {
30. NumberChanger n1 = new NumberChanger(AddNum);
31. NumberChanger n2 = new NumberChanger(MultNum);
32. n1(25);
33. Console.WriteLine("Value of Num: {0}", getNum());
34. n2(5);
35. Console.WriteLine("Value of Num: {0}", getNum());
36. Console.ReadKey();
37. }
38. }
39. }

advertisements

Here is the output of the C# Program:

Value of Num: 35

Value of Num: 175

C# Program to Use Delegate to Call 2 Methods within which First method Prints to Console and Second Method Prints to File

This C# Program to Use Delegate to Call 2 Methods within which First method Prints to Console and Second Method Prints to File. Here it uses delegates to create two methods one to print to the console and other to file.

Here is source code of the C# Program to Use Delegate to Call 2 Methods within which First method Prints to Console and Second Method Prints to File. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Use Delegate to Call 2 Methods within which First method Prints to Console and Second Method Prints to File*
3. *\*/*
4. **using** System;
5. **using** System.IO;
6. **namespace** Program
7. {
8. **class** PrintString
9. {
10. **static** FileStream fs;
11. **static** StreamWriter sw;
12. **public** **delegate** **void** printString(**string** s);
13. **public** **static** **void** Screen(**string** str)
14. {
15. Console.WriteLine("The String is: {0}", str);
16. }
17. **public** **static** **void** File(**string** s)
18. {
19. fs = new FileStream("c:**\\**sri**\\**Message.txt",
20. FileMode.Append, FileAccess.Write);
21. sw = new StreamWriter(fs);
22. sw.WriteLine(s);
23. sw.Flush();
24. sw.Close();
25. fs.Close();
26. }
27. **public** **static** **void** sendString(printString ps)
28. {
29. ps("C# Program to Use Delegate to Call 2 Methods within which First method Prints to Console and Second Method Prints to File");
30. }
31. **static** **void** Main(**string**[] args)
32. {
33. printString ps1 = new printString(Screen);
34. printString ps2 = new printString(File);
35. sendString(ps1);
36. sendString(ps2);
37. Console.ReadKey();
38. }
39. }
40. }

advertisements

Here is the output of the C# Program:

The String is : C# Program to Use Delegate to Call 2 Methods within which First method Prints to Console and Second Method Prints to File

#### I. C# Programming Examples on Events

The C# programs here illustrate mouse handling events, elapsed events, clones, predicates and actions and displaying time for different regions.

#### 1. C# Examples on Mouse Handling Events

In the MOUSEUP Event, when the mouse is moved up over the button message box is displayed in windows forms application. The KEYUP event is generated when the user releases a key on the keyboard. TimeSpan represents a certain time period. The following programs perform addition and subtraction with MOUSEUP Event and KEYUP Event and adds two TimeSpans.

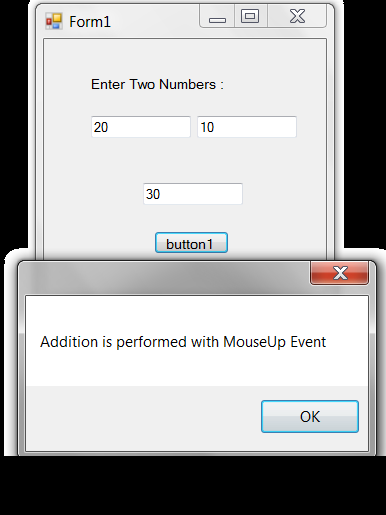
C# Program to Perform Addition with MOUSEUP Event

This C# Program Performs Addition with MOUSEUP Event. Here when the mouse is moved up over the button message box is displayed in windows forms application.

Here is source code of the C# Program to Perform Addition with MOUSEUP Event. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Perform Addition with MOUSEUP Event*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.ComponentModel;
7. **using** System.Data;
8. **using** System.Drawing;
9. **using** System.Linq;
10. **using** System.Text;
11. **using** System.Windows.Forms;
13. **namespace** WindowsFormsApplication14
14. {
15. **public** **partial** **class** Form1 : Form
16. {
17. **public** Form1()
18. {
19. InitializeComponent();
20. }
22. **private** **void** button1\_MouseUp(**object** sender, MouseEventArgs e)
23. {
24. **int** **add**;
25. **add** = Convert.ToInt32(textBox1.Text) +Convert.ToInt32(textBox2.Text);
26. textBox3.Text = Convert.ToString(**add**);
27. MessageBox.Show("Addition is performed with MouseUp Event");
28. }
29. }
30. }

advertisements

Here is the output of the C# Program:  


C# Program to Add Two TimeSpan

This C# Program Adds Two TimeSpan . Here the two timespans are added and the value is displayed.

Here is source code of the C# Program to Add Two TimeSpan .The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Add Two TimeSpan*
3. *\*/*
4. **using** System;
5. **class** Program
6. {
7. **static** **void** Main()
8. {
9. TimeSpan s1 = new TimeSpan(5, 0, 0);
10. TimeSpan s2 = new TimeSpan(2, 0, 0);
11. TimeSpan s3 = s1.**Add**(s2);
12. Console.WriteLine(s3);
13. Console.Read();
14. }
15. }

Here is the output of the C# Program:

07:00:00

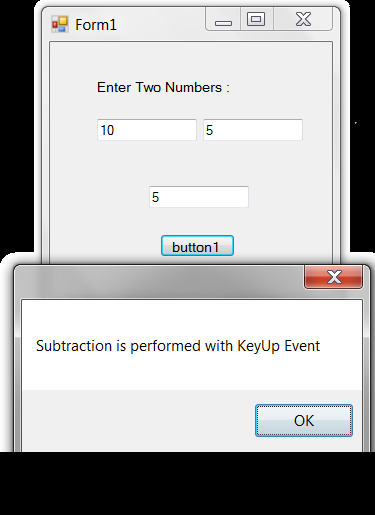
C# Program to Perform Subtraction with Key Up Event

This C# Program Performs Subtraction with Key Up Event. Here keyup event Fires when the user releases a key, after the default action of that key has been performed.

Here is source code of the C# Program to Perform Subtraction with Key Up Event. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Perform Subtraction with Key Up Event*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.ComponentModel;
7. **using** System.Data;
8. **using** System.Drawing;
9. **using** System.Linq;
10. **using** System.Text;
11. **using** System.Windows.Forms;
13. **namespace** WindowsFormsApplication14
14. {
15. **public** **partial** **class** Form1 : Form
16. {
17. **public** Form1()
18. {
19. InitializeComponent();
20. }
22. **private** **void** button1\_KeyUp(**object** sender, KeyEventArgs e)
23. {
24. **int** sub;
25. sub = Convert.ToInt32(textBox1.Text) - Convert.ToInt32(textBox2.Text);
26. textBox3.Text = Convert.ToString(sub);
27. MessageBox.Show("Subtraction is performed with KeyUp Event");
29. }
30. }
31. }

advertisements

Here is the output of the C# Program:  


#### 2. C# Examples on Displaying the Times

With universal times, you can compare times from different locations with no errors. The following Programs determine the day time and universal time of various locations. The programs also determine if the enter key is pressed or not, gets the day and the DayLight saving information.

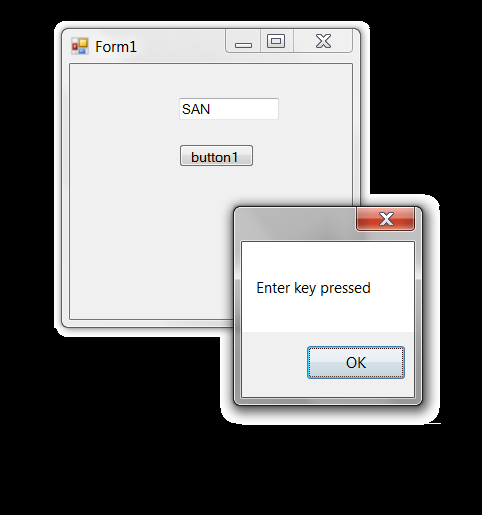
C# Program to Detect Whether ENTER key is Pressed or Not

This C# Program to Detect Whether ENTER key is Pressed or Not. Here when the enter key is pressed message box is displayed in windows forms application.

Here is source code of the C# Program to Detect Whether ENTER key is Pressed or Not. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Detect Whether ENTER key is Pressed or Not*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.ComponentModel;
7. **using** System.Data;
8. **using** System.Drawing;
9. **using** System.Linq;
10. **using** System.Text;
11. **using** System.Windows.Forms;
13. **namespace** WindowsFormsApplication12
14. {
15. **public** **partial** **class** Form1 : Form
16. {
17. **public** Form1()
18. {
19. InitializeComponent();
20. }
21. **private** **void** textBox1\_KeyUp(**object** sender, KeyEventArgs e)
22. {
23. **if** (e.KeyValue == 13)
24. MessageBox.Show("Enter key pressed");
25. }
26. }
27. }

advertisements

Here is the output of the C# Program:  


C# Program to get the Universal Time

This C# Program to get the Universal Time. Here the universal time is calculated using the timezone function.

Here is source code of the C# Program to get the Universal Time. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to get the Universal Time*
3. *\*/*
4. **using** System;
5. **class** Program
6. {
7. **static** **void** Main()
8. {
9. TimeZone zone = TimeZone.CurrentTimeZone;
10. DateTime univ = zone.ToUniversalTime(DateTime.Now);
11. Console.WriteLine("Universal Time is {0}",univ);
12. Console.Read();
13. }
14. }

advertisements

Here is the output of the C# Program:

Universal Time is 10/22/2013 5:23:25 PM

C# Program to get the Local Time

This C# Program to get the Local Time. Here the local time is calculated using timezone function.

Here is source code of the C# Program to get the Local Time. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to get the Local Time*
3. *\*/*
4. **using** System;
5. **class** Program
6. {
7. **static** **void** Main()
8. {
9. TimeZone zone = TimeZone.CurrentTimeZone;
10. DateTime local = zone.ToLocalTime(DateTime.Now);
11. Console.WriteLine("The Local Time is : {0}",local);
12. Console.ReadLine();
13. }
14. }

advertisements

Here is the output of the C# Program:

The Local Time is : 10/22/2013 11:01:51 PM

C# Program to Get the DayLight Saving Information

This C# Program Gets the DayLight Saving Information. Here the information about the current year’s daylight saving changes is obtained.

Here is source code of the C# Program to Get the DayLight Saving Information. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Get the DayLight Saving Information*
3. *\*/*
4. **using** System;
5. **using** System.Globalization;
6. **class** Program
7. {
8. **static** **void** Main()
9. {
10. TimeZone z = TimeZone.CurrentTimeZone;
11. DaylightTime t = z.GetDaylightChanges(DateTime.Today.Year);
12. Console.WriteLine("Start Time: {0}", t.Start);
13. Console.WriteLine("Delta Time: {0}", t.Delta);
14. Console.WriteLine("End Time: {0}", t.End);
15. Console.ReadLine();
16. }
17. }

advertisements

Here is the output of the C# Program:

Start Time: 1/1/0001 12:00:00 AM

Delta Time: 00:00:00

End Time: 1/1/0001 12:00:00 AM

#### 3. C# Examples on Predicates and Actions

A predicate delegate is a delegate which is a boolean function that returns true or false and takes a generic type as an argument. Radio buttons are used to select just one option from a set of alternatives. The following programs illustrates Predicates and Actions, creates a Progress Bar Control, creates Input Box and displays the text and creates a Radio Button and demonstrates its use.

C# Program to Illustrate Predicate

This C# Program Illustrates Predicate. Here Predicate generic delegate type with an int type parameter and it returns true if the argument passed matches with the value.

Here is source code of the C# Program to Illustrate Predicate. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Illustrate Predicate*
3. *\*/*
4. **using** System;
6. **class** Program
7. {
8. **static** **void** Main()
9. {
10. Predicate<**int**> predicate = checkval => checkval == 6;
11. Console.WriteLine(predicate.Invoke(4));
12. Console.WriteLine(predicate.Invoke(5));
13. Console.WriteLine(predicate.Invoke(6));
14. Console.Read();
15. }
16. }

advertisements

Here is the output of the C# Program:

False False True

C# Program to Illustrate Actions

This C# Program Illustrates Actions. Here Action objects return no values.

Here is source code of the C# Program to Illustrate Actions. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Illustrate Actions*
3. *\*/*
4. **using** System;
6. **class** Program
7. {
8. **static** **void** Main()
9. {
10. Action<**int**> action1 =(**int** x) => Console.WriteLine("OUTPUT {0}", x);
11. Action<**int**, **int**> action2 =(x, y) => Console.WriteLine("OUTPUT {0} and {1}", x, y);
12. action1.Invoke(1111);
13. action2.Invoke(200, 3000);
14. Console.Read();
15. }
16. }

advertisements

Here is the output of the C# Program:

OUTPUT 1111

OUTPUT 200 and 3000

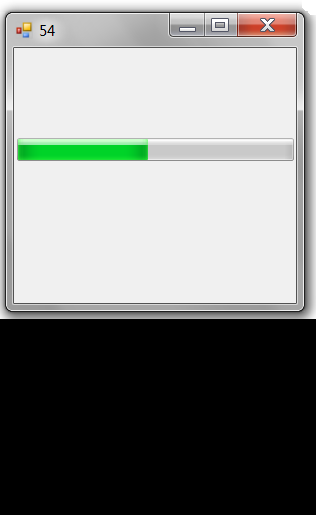
C# Program to Create a Progress Bar Control

This C# Program Creates a Progress Bar Control. Here ProgressBar indicates visually the progress of an operation. It is best used on a long-running computation or task.

Here is source code of the C# Program to Create a Progress Bar Control. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Create a Progress Bar Control*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.ComponentModel;
7. **using** System.Data;
8. **using** System.Drawing;
9. **using** System.Linq;
10. **using** System.Text;
11. **using** System.Windows.Forms;
13. **namespace** ProgressBar
14. {
15. **public** **partial** **class** Form1 : Form
16. {
17. **public** Form1()
18. {
19. InitializeComponent();
20. }
22. **private** **void** backgroundWorker1\_DoWork(**object** sender, DoWorkEventArgs e)
23. {
24. **for** (**int** i = 1; i <= 100; i++)
25. {
26. System.Threading.Thread.Sleep(100);
27. backgroundWorker1.ReportProgress(i);
28. }
29. }
31. **private** **void** backgroundWorker1\_ProgressChanged(**object** sender, ProgressChangedEventArgs e)
32. {
33. progressBar1.**Value** = e.ProgressPercentage;
34. **this**.Text = e.ProgressPercentage.ToString();
35. }
37. **private** **void** Form1\_Load(**object** sender, EventArgs e)
38. {
39. backgroundWorker1.RunWorkerAsync();
40. }
41. }
42. }

advertisements

Here is the output of the C# Program:  


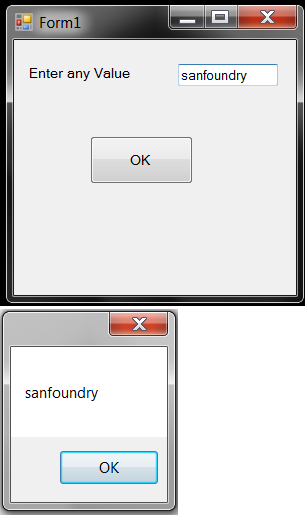
C# Program to Create Input Box and Display the Text

This C# Program Creates Input Box and Display the Text. Here the Input box is created and the text which is entered in the text box is displayed using the message box in windows forms application.

Here is source code of the C# Program to Create Input Box and Display the Text. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Create Input Box and Display the Text*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.ComponentModel;
7. **using** System.Data;
8. **using** System.Drawing;
9. **using** System.Linq;
10. **using** System.Text;
11. **using** System.Windows.Forms;
13. **namespace** WindowsFormsApplication6
14. {
15. **public** **partial** **class** Form1 : Form
16. {
17. **public** Form1()
18. {
19. InitializeComponent();
21. }
23. **private** **void** button1\_Click(**object** sender, EventArgs e)
24. {
25. MessageBox.Show(textBox1.Text);
26. Console.ReadLine();
27. }
28. }
29. }

advertisements

Here is the output of the C# Program:  


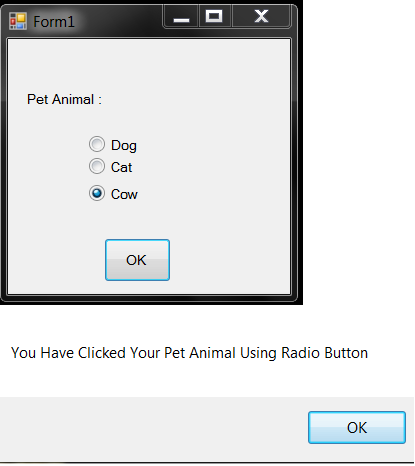
C# Program to Create Radio Button and Demonstrate its Use

This C# Program Creates Radio Button and Demonstrate its Use. Here the radio button is created with the windows forms application and the code is written in c#.

Here is source code of the C# Program to Create Radio Button and Demonstrate its Use. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Create Radio Button and Demonstrate its Use*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.ComponentModel;
7. **using** System.Data;
8. **using** System.Drawing;
9. **using** System.Linq;
10. **using** System.Text;
11. **using** System.Windows.Forms;
12. **namespace** WindowsFormsApplication5
13. {
14. **public** **partial** **class** Form1 : Form
15. {
16. **public** Form1()
17. {
18. InitializeComponent();
19. }
21. **private** **void** label1\_Click(**object** sender, EventArgs e)
22. {
24. }
26. **private** **void** button1\_Click(**object** sender, EventArgs e)
27. {
28. MessageBox.Show("You Have Clicked Your Pet Animal Using Radio Button ");
29. }
30. }
31. }

advertisements

Here is the output of the C# Program:  


#### 4. C# Examples on Elapsed Events and Clones

Event handlers attached with .on() are triggered when the corresponding event occurs. C# Clone() method creates and returns a copy of the object. The Elapsed event occurs if the Enabled property is true and the time interval defined by the interval property elapses. The following programs illustrate elapsed event, demonstrates the use of clone, demonstrates the concept of trigger and creates a stop watch.

C# Program to Illustrate Elapsed Event

This C# Program Illustrates Elapsed Event. Here an Event Handler is set up for the Timer.Elapsed event, creates a timer, and starts the timer. The event handler displays the SignalTime property each time it is raised.

Here is source code of the C# Program to Illustrate Elapsed Event. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Illustrate Elapsed Event*
3. *\*/*
4. **using** System;
5. **using** System.Timers;
6. **public** **class** Program
7. {
8. **private** **static** System.Timers.Timer Tim;
9. **public** **static** **void** Main()
10. {
11. Tim = new System.Timers.Timer(10);
12. Tim.Elapsed += new ElapsedEventHandler(OnTimedEvent);
13. Tim.Interval = 1000;
14. Tim.Enabled = **true**;
15. Console.WriteLine("Press Any Key to Exit else Elapsed Event will be Raised ");
16. Console.ReadLine();
17. }
18. **private** **static** **void** OnTimedEvent(**object** source, ElapsedEventArgs e)
19. {
20. Console.WriteLine("The Elapsed event was Raised {0}", e.SignalTime);
21. }
22. }

advertisements

Here is the output of the C# Program:

Press Any Key to Exit else Elapsed Event will be Raised :

The Elapsed event was raised at 9/17/2013 7:24:15 PM

The Elapsed event was raised at 9/17/2013 7:24:16 PM

The Elapsed event was raised at 9/17/2013 7:24:17 PM

C# Program to Demonstrate Use of Clone

This C# Program Demonstrates Use of Clone. Here the cloning is done and displayed.

Here is source code of the C# Program to Demonstrate Use of Clone. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Demonstrate Use of Clone*
3. *\*/*
4. **using** System;
5. **class** Program
6. {
7. **static** **void** Main()
8. {
9. **string**[] names = { "San", "Csharp", "Linux" };
10. **string**[] clonenames = names.Clone() **as** **string**[];
11. Console.WriteLine(**string**.**Join**(",", names));
12. Console.WriteLine(**string**.**Join**(",", clonenames));
13. Console.WriteLine();
14. clonenames[0] = "SanFoundry";
15. Console.WriteLine(**string**.**Join**(",", names));
16. Console.WriteLine(**string**.**Join**(",", clonenames));
17. Console.Read();
18. }
19. }

advertisements

Here is the output of the C# Program:

San,Csharp,Linux

San,Csharp,Linux

San,Csharp,Linux

SanFoundry,Csharp,Linux

C# Program to Demonstrate Trigger Concept

This C# Program Demonstrates Trigger Concept. Here the trigger concept is used and the numbers are added and displayed.

Here is source code of the C# Program to Demonstrate Trigger Concept. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Demonstrate Trigger Concept*
3. *\*/*
4. **using** System;
5. **delegate** **bool** Condition(**object** obj);
6. **delegate** **void** Action(**object** obj);
7. **class** Counter
8. {
9. **int** val = 0;
11. **public** **event** Condition cond;
12. **public** **event** Action evn;
14. **public** **int** **Value** { **get** { **return** val; } }
16. **public** **void** addition(**int** x)
17. {
18. val += x; Checkpoint();
19. }
21. **public** **void** Clearall()
22. {
23. val = 0; Checkpoint();
24. }
26. **void** Checkpoint()
27. {
28. **if** (cond != **null** && evn != **null** && cond(**this**)) evn(**this**);
29. }
30. }
31. **class** Test
32. {
33. **static** **int** hval = 0;
34. **static** **bool** CheckpointLimit(**object** ctr)
35. {
36. **return** (((Counter)ctr).**Value** > 100);
37. }
38. **static** **void** Alarm(**object** ctr)
39. {
40. Console.WriteLine("Counter Overflow");
41. }
42. **static** **void** Reset(**object** ctr)
43. {
44. hval = ((Counter)ctr).**Value**;
45. Console.WriteLine("hval = " + hval);
46. ((Counter)ctr).Clearall();
47. }
48. **public** **static** **void** Main()
49. {
50. Counter counter = new Counter();
51. counter.cond += new Condition(CheckpointLimit);
52. counter.evn += new Action(Alarm);
53. counter.evn += new Action(Reset);
54. counter.addition(10);
55. counter.addition(20);
56. counter.addition(30);
57. counter.addition(40);
58. counter.addition(50);
59. Console.Read();
60. }
61. }

advertisements

Here is the output of the C# Program:

Counter Overflow

hval = 150

C# Program to Create Stop Watch

This C# Program Creates Stop Watch .Here the stop watch is created and the time for the printing function is calculated and they are displayed.

Here is source code of the C# Program to Create Stop Watch. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Create Stop Watch*
3. *\*/*
4. **using** System;
5. **using** System.Diagnostics;
6. **using** System.Threading;
7. **class** Program
8. {
9. **static** **void** Main()
10. {
11. *// Create new stopwatch*
12. Stopwatch stopwatch = new Stopwatch();
13. *// Begin timing*
14. stopwatch.Start();
15. **for** (**int** i = 0; i < 10; i++)
16. {
17. Console.WriteLine("HI");
18. }
19. *// Stop timing*
20. stopwatch.Stop();
21. Console.WriteLine("Time Elapsed : {0}",
22. stopwatch.Elapsed);
23. Console.ReadLine();
24. }
25. }

advertisements

Here is the output of the C# Program:

HI

HI

HI

HI

HI

HI

HI

HI

HI

HI

Time Elapsed : 00:00:00.0015114

#### J. C# Programming Examples on Exceptions

The C# programs in this section demonstrates various aspects of exception handling.

#### C# Examples demonstrating various aspects of Exception Handling

An event that occurs during the program execution, which disrupts the normal flow of the program is called an Exception. IndexOutOfRange exception is thrown when an attempt is made to access an element of an array with an index that is outside the bounds of the array. Divide By Zero Exception is Thrown when an exceptional arithmetic condition has occurred. The programs in the section illustrate IndexOutOfRange Exception and DivideByZero Exception. The following programs illustrate exception handling for Invalid TypeCasting in unBoxing and demonstrate exception handling for stack overflow. A NullReferenceException happens when you try to access a reference variable that isn’t referencing any object. The remaining programs in the section demonstrate multiple exceptions and illustrate NullRefernce Exception.

C# Program to Demonstrate IndexOutOfRange Exception

This C# Program Demonstrates IndexOutOfRange Exception. Here if the array has the index value out of the range that is specified then this exception is thrown.

Here is source code of the C# Program to Demonstrate IndexOutOfRange Exception. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Demonstrate IndexOutOfRange Exception*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
8. **namespace** differnce
9. {
10. **class** arrayoutofindex
11. {
12. **public** **void** calculatedifference()
13. {
14. **int** difference=0;
15. **int** [] number= new **int**[5] {1,2,3,4,5};
16. **try**
17. {
18. **for** (**int** init =1; init <=5; init++)
19. {
20. difference= difference - number[init];
21. }
22. Console.WriteLine("The difference of the array is:" + difference);
23. }
24. **catch** (IndexOutOfRangeException e)
25. {
26. Console.WriteLine(e.Message);
27. }
28. }
29. }
30. **class** classmain
31. {
32. **static** **void** Main(**string** [] args)
33. {
34. arrayoutofindex obj = new arrayoutofindex();
35. obj.calculatedifference();
36. Console.ReadLine();
37. }
38. }
39. }

advertisements

Here is the output of the C# Program:

Index was outside the bounds of the array

C# Program to Demonstrate DivideByZero Exception

This C# Program Demonstrates DivideByZero Exception. Here A DivideByZeroException is thrown. It indicates that a statement attempted to evaluate a division by zero.

Here is source code of the C# Program to Demonstrate DivideByZero Exception. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Demonstrate DivideByZero Exception*
3. *\*/*
4. **using** System;
5. **class** Program
6. {
7. **static** **void** Main()
8. {
9. **try**
10. {
11. **int** result = 15 / **int**.Parse("0");
12. Console.WriteLine(result);
13. }
14. **catch** (DivideByZeroException e)
15. {
16. Console.Write(e.Message);
17. Console.ReadLine();
18. }
19. }
20. }

advertisements

Here is the output of the C# Program:

Attempted to Divide by Zero

C# Program to Illustrate Exception Handling for Invalid TypeCasting in UnBoxing

This C# Program Illustrates Exception Handling for Invalid TypeCasting in UnBoxing. Here it demonstrates a case of invalid unboxing and the resulting InvalidCastException. Using try and catch, an error message is displayed when the error occurs.

Here is source code of the C# Program to Illustrate Exception Handling for Invalid TypeCasting in UnBoxing. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Illustrate Exception Handling for Invalid TypeCasting in UnBoxing*
3. *\*/*
4. **class** TestUnboxing
5. {
6. **static** **void** Main()
7. {
8. **int** num = 123;
9. **object** obj = num;
10. **try**
11. {
12. **int** j = (**short**)obj;
13. System.Console.WriteLine("Unboxing");
14. }
15. **catch** (System.InvalidCastException e)
16. {
17. System.Console.WriteLine("{0} Error: Incorrect unboxing", e.Message);
18. }
19. System.Console.Read();
20. }
21. }

advertisements

Here is the output of the C# Program:

Specified Cast is not Valid.Error : Incorrect Boxing

C# Program to Demonstrate Multiple Exceptions

This C# Program Demonstrates Multiple Exceptions. Here Exceptions in C# provide a structured, uniform, and type-safe way of handling both system-level and application-level error conditions..

Here is source code of the C# Program to Demonstrate Multiple Exceptions. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Demonstrate Multiple Exceptions*
3. *\*/*
4. **using** System;
5. **class** Exercise
6. {
7. **static** **void** Main()
8. {
9. **double** Num1, Num2;
10. **double** Result = 0.00;
11. **char** op;
12. **try**
13. {
14. Console.Write("Enter your First Number : ");
15. Num1 = **double**.Parse(Console.ReadLine());
16. Console.Write("Enter an Operator (+, -, \* or /): ");
17. op = **char**.Parse(Console.ReadLine());
18. **if** (op != '+' && op != '-' &&
19. op != '\*' && op != '/')
20. **throw** new Exception(op.ToString());
21. Console.Write("Enter your Second Number :");
22. Num2 = **double**.Parse(Console.ReadLine());
23. **if** (op == '/')
24. **if** (Num2 == 0)
25. **throw** new DivideByZeroException("Division by zero is not allowed");
26. Result = Calculator(Num1, Num2, op);
27. Console.WriteLine("**\n**{0} {1} {2} = {3}", Num1, op, Num2, Result);
28. }
29. **catch** (FormatException)
30. {
31. Console.WriteLine("The number you typed is not valid");
32. }
33. **catch** (DivideByZeroException ex)
34. {
35. Console.WriteLine(ex.Message);
36. }
37. **catch** (Exception ex)
38. {
39. Console.WriteLine("Operation Error: {0} is not a valid op", ex.Message);
40. }
41. Console.Read();
42. }
44. **static** **double** Calculator(**double** v1, **double** v2, **char** op)
45. {
46. **double** Result = 0.00;
48. **switch** (op)
49. {
50. **case** '+':
51. Result = v1 + v2;
52. **break**;
53. **case** '-':
54. Result = v1 - v2;
55. **break**;
56. **case** '\*':
57. Result = v1 \* v2;
58. **break**;
59. **case** '/':
60. Result = v1 / v2;
61. **break**;
62. }
63. **return** Result;
64. }
65. }

advertisements

Here is the output of the C# Program:

Enter Your First Number : 10

Enter an Operator (+, -, \* or /) : ,

Operation Error : , is not a Valid Operator

C# Program to Demonstrate Exception Handling for Stack Overflow

This C# Program Demonstrates Exception Handling for Stack Overflow. Here Typically the StackOverflowException is triggered by a recursive method that creates a deep call stack.

Here is source code of the C# Program to Demonstrate Exception Handling for Stack Overflow. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Demonstrate Exception Handling for Stack Overflow*
3. *\*/*
4. **using** System;
5. **class** Program
6. {
7. **static** **void** excep(**int** **value**)
8. {
9. Console.WriteLine(**value**);
10. excep(++**value**);
11. }
13. **static** **void** Main()
14. {
15. **try**
16. {
17. excep(0);
18. }
19. **catch** (StackOverflowException e)
20. {
21. Console.WriteLine(e.Message);
22. }
23. }
24. }

Here is the output of the C# Program:

Program terminated due to StackOverflow Exception.

C# Program to Illustrate NullRefernce Exception

This C# Program Illustrates NullRefernce Exception. Here It indicates that you are trying to access member fields, or function types, on an object reference that points to null.

Here is source code of the C# Program to Illustrate NullRefernce Exception. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Illustrate NullRefernce Exception*
3. *\*/*
4. **using** System;
5. **class** Program
6. {
7. **static** **void** Main()
8. {
9. **try**
10. {
11. **string** **value** = **null**;
12. **if** (**value**.Length == 0)
13. {
14. Console.WriteLine(**value**);
15. }
16. }
17. **catch**(NullReferenceException e)
18. {
19. Console.WriteLine(e.Message);
20. }
21. Console.Read();
22. }
23. }

advertisements

Here is the output of the C# Program:

Object Reference not set to an Instance of an Object

#### K. C# Programming Examples on Gaming

The programs in this section provides demonstration of various games.

#### C# Examples on Different Games

This section contains various programs on Gaming. They are Hangman Game, Tower of Hanoi, Number Guessing Game and Prefix Game. HangMan Game is a game which consists of two or more players. One of the players tells a word or a sentence and the other player tries to guess that word by providing various suggestions. The Tower of Hanoi problem consists of three towers and a number of disks. The objective is to place all the disks on Tower 3 such that the larger disk cannot be on the smaller one. In the Number Guessing Game, there are 2 buttons, Yes and No. First think of a number in your mind. The computer displays all the positive integers less than 1000 on the screen. If the number you have thought of appears on the screen press yes. After a while the computer will guess your number. In the Prefix Game, prefices are added to the words. Please find the programs below:

C# Program to Create a HangMan Game

This C# Program Creates a HangMan Game. Here words are guessed by the user and the game is continued.

Here is source code of the C# Program to Create a HangMan Game. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Create a HangMan Game*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
9. **namespace** Hangman
10. {
11. **class** Program
12. {
13. **static** **void** Main(**string**[] args)
14. {
16. Console.WriteLine("Welcome to Hangman!!!!!!!!!!");
17. **string**[] listwords = new **string**[10];
18. listwords[0] = "sheep";
19. listwords[1] = "goat";
20. listwords[2] = "computer";
21. listwords[3] = "america";
22. listwords[4] = "watermelon";
23. listwords[5] = "icecream";
24. listwords[6] = "jasmine";
25. listwords[7] = "pineapple";
26. listwords[8] = "orange";
27. listwords[9] = "mango";
28. Random randGen = new Random();
29. **var** idx = randGen.Next(0, 9);
30. **string** mysteryWord = listwords[idx];
31. **char**[] guess = new **char**[mysteryWord.Length];
32. Console.Write("Please enter your guess: ");
34. **for** (**int** p = 0; p < mysteryWord.Length; p++)
35. guess[p] = '\*';
37. **while** (**true**)
38. {
39. **char** playerGuess = **char**.Parse(Console.ReadLine());
40. **for** (**int** j = 0; j < mysteryWord.Length; j++)
41. {
42. **if** (playerGuess == mysteryWord[j])
43. guess[j] = playerGuess;
44. }
45. Console.WriteLine(guess);
46. }
47. }
48. }
49. }

advertisements

Here is the output of the C# Program:

Welcome to Hangman!!!!!!!!!!

Please enter your guess: i

\*\*\*\*

a

\*\*a\*

e

\*\*a\*

g

g\*a\*

o

goa\*

t

goat

C# Program to Demonstrate Tower Of Hanoi

This C# Program uses recursive function & solves the tower of hanoi. The tower of hanoi is a mathematical puzzle. It consists of threerods, and a number of disks of different sizes which can slideonto any rod. The puzzle starts with the disks in a neat stack in ascending order of size on one rod, the smallest at the top. We have to obtain the same stack on the third rod..

Here is source code of the C# Program to Demonstrate Tower Of Hanoi. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Demonstrate Tower Of Hanoi*
3. *\*/*
4. **using** System;
5. **class** TowerOfHanoi
6. {
7. **int** m\_numdiscs;
8. **public** TowerOfHanoi()
9. {
10. numdiscs = 0;
11. }
12. **public** TowerOfHanoi(**int** newval)
13. {
14. numdiscs = newval;
15. }
16. **public** **int** numdiscs
17. {
18. **get**
19. {
20. **return** m\_numdiscs;
21. }
22. **set**
23. {
24. **if** (**value** > 0)
25. m\_numdiscs = **value**;
26. }
27. }
28. **public** **void** movetower(**int** n, **int** **from**, **int** to, **int** other)
29. {
30. **if** (n > 0)
31. {
32. movetower(n - 1, **from**, other, to);
33. Console.WriteLine("Move disk {0} from tower {1} to tower {2}", n, **from**, to);
34. movetower(n - 1, other, to, **from**);
35. }
36. }
37. }
38. **class** TowersOfHanoiApp
39. {
40. **public** **static** **int** Main()
41. {
42. TowerOfHanoi T = new TowerOfHanoi();
43. **string** cnumdiscs;
44. Console.Write("Enter the number of discs: ");
45. cnumdiscs = Console.ReadLine();
46. T.numdiscs = Convert.ToInt32(cnumdiscs);
47. T.movetower(T.numdiscs, 1, 3, 2);
48. Console.ReadLine();
49. **return** 0;
50. }
51. }

advertisements

Here is the output of the C# Program:

Enter the Number of Disks : 2

Move Disk 1 from Tower 1 to Tower 2

Move Disk 1 from Tower 1 to Tower 3

Move Disk 1 from Tower 2 to Tower 3

C# Program to Perform a Number Guessing Game

This C# Program Performs a Number Guessing Game. Here a Simple number guessing game is created.

Here is source code of the C# Program to Perform a Number Guessing Game. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Perform a Number Guessing Game*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Text;
7. **class** Program
8. {
9. **static** **void** Main(**string**[] args)
10. {
11. **while** (**true**)
12. {
13. **int** randno = Newnum(1, 101);
14. **int** count = 1;
15. **while** (**true**)
16. {
17. Console.Write("Enter a number between 1 and 100(0 to quit):");
18. **int** input = Convert.ToInt32(Console.ReadLine());
19. **if** (input == 0)
20. **return**;
21. **else** **if** (input < randno)
22. {
23. Console.WriteLine("Low, try again.");
24. ++count;
25. **continue**;
26. }
27. **else** **if** (input > randno)
28. {
29. Console.WriteLine("High, try again.");
30. ++count;
31. **continue**;
32. }
33. **else**
34. {
35. Console.WriteLine("You guessed it! The number was {0}!", randno);
36. Console.WriteLine("It took you {0} {1}.**\n**", count, count == 1 ? "try" : "tries");
37. **break**;
38. }
39. }
40. }
42. }
43. **static** **int** Newnum(**int** min, **int** max)
44. {
45. Random random = new Random();
46. **return** random.Next(min, max);
47. }
48. }
49. }

advertisements

Here is the output of the C# Program:

Enter a number between 1 and 100(0 to quit) : 56

Low,try again.

Enter a number between 1 and 100(0 to quit): 67

high,try again.

Enter a number between 1 and 100(0 to quit): 59

You guessed it! The number was 59

It took you 2 tries!!!

C# Program to Prefix Game

This C# Program to Prefix Game. Here prefix has to be found for the words which are given.

Here is source code of the C# Program to Prefix Game. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Prefix Game*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
8. **namespace** Project
9. {
10. **class** Program
11. {
12. **static** **void** Main(**string**[] args)
13. {
14. **string**[,] seq = new **string**[4, 3];
15. **int** row = 0;
16. seq[row, 0] = "substring";
17. seq[row, 1] = "sub";
18. seq[row, 2] = "incorrect";
19. row++;
20. seq[row, 0] = "input";
21. seq[row, 1] = "in";
22. seq[row, 2] = "incorrect";
23. row++;
24. **int** numrows = 0;
25. play(seq, numrows);
26. Console.Write("Continue : Press 'y'");
27. **string** next = Console.ReadLine();
28. **if** (next.CompareTo("y") == 0)
29. {
30. Console.Clear();
31. numrows += 2;
32. play(seq, numrows);
33. }
34. Console.ReadLine();
35. }
36. **static** **void** play(**string**[,] seq, **int** rows)
37. {
38. Console.WriteLine("ENGLISH WORD PREFIX GAME");
39. **for** (**int** i = rows; i <rows+2 ; i++)
40. {
41. Console.Write("What is the correct prefix of '{0}':", seq[i, 0]);
42. **string** ans = Console.ReadLine();
43. **if** (seq[i, 1].ToLower().CompareTo(ans.ToString().ToLower()) == 0)
44. seq[i, 2] = "correct";
45. Console.WriteLine();
46. }
47. Console.WriteLine("CHECK YOUR ANSWERS!!!");
48. Console.WriteLine("Word**\t\t\t**Prefix**\t\t**Description");
49. **for** (**int** i = rows; i < rows+2; i++)
50. {
51. **for** (**int** j = 0; j < 3; j++)
52. Console.Write("{0}**\t\t**", seq[i, j]);
53. Console.WriteLine();
54. }
55. Console.Read();
56. }
57. }
58. }

advertisements

Here is the output of the C# Program:

ENGLISH WORD PREFIX GAME

What is the correct prefix of 'substring' : sub

What is the correct prefix of 'input' : in

CHECK YOUR ANSWERS!!!!

Word Prefix Description

substring sub correct

input in correct

#### L. C# Programming Examples on Files

The programs in this section illustrate files, directories, StringReader, StringWriter, StringBuilder, Stream Reader, Binary Reader and various other file functions.

#### 1. C# Examples on Files and Directories

A file is a collection of data and information and a Directory is a location in the computer where files are stored. The programs in the given section creates a file, reads the contents of a file, creates a directory, views the information of the file, views the date and time of access of a file and checks the existence of a file.

C# Program to Create a File

This C# Program Creates a File. Here the file is created and the content is written to the file and the same content is displayed.

Here is source code of the C# Program to Create a File. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Create a File*
3. *\*/*
4. **using** System;
5. **using** System.IO;
6. **using** System.Text;
7. **class** Test
8. {
9. **public** **static** **void** Main()
10. {
11. **string** textpath = @"c:\sri\test.txt";
12. **using** (FileStream fs = File.Create(textpath))
13. {
14. **Byte**[] info = new UTF8Encoding(**true**).GetBytes("File is Created");
15. fs.Write(info, 0, info.Length);
16. }
17. **using** (StreamReader sr = File.OpenText(textpath))
18. {
19. **string** s = "";
20. **while** ((s = sr.ReadLine()) != **null**)
21. {
22. Console.WriteLine(s);
23. }
24. }
25. Console.Read();
26. }
27. }

advertisements

Here is the output of the C# Program:

File is Created

C# Program to Read the Contents of the File

This C# Program Reads the Contents of a File.It uses the library functions to read the data from the file that is created already in the same path of the program that is written.

Here is source code to Read the Contents of a File.The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Read Contents of a File*
3. *\*/*
4. **using** System;
5. **using** System.IO;
6. **class** FileRead
7. {
8. **public** **void** readdata()
9. {
10. FileStream fs = new FileStream("Myfile.txt", FileMode.Open, FileAccess.Read);
11. StreamReader sr = new StreamReader(fs);*//Position the File Pointer at the Beginning of the File*
12. sr.BaseStream.Seek(0, SeekOrigin.Begin);*//Read till the End of the File is Encountered*
13. **string** str = sr.ReadLine();
14. **while** (str != **null**)
15. {
16. Console.WriteLine("{0}", str);
17. str = sr.ReadLine();
18. }
19. *//Close the Writer and File*
20. sr.Close();
21. fs.Close();
22. }
23. **public** **static** **void** Main(**String**[] args)
24. {
25. FileRead fr = new FileRead();
26. fr.readdata();
27. }
28. }

Here is the output of the C# Program:

The text which your are reading are read from the file named myfile.txt that is created already.

C# Program to Create a Directory

This C# Program Creates a Directory. Here a new Directory is created in C directory using createdirectory function.

Here is source code of the C# Program to Create a Directory. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Create a Directory*
3. *\*/*
4. **using** System;
5. **using** System.IO;
6. **class** program
7. {
8. **public** **static** **void** Main()
9. {
10. Directory.CreateDirectory("C:**\\**NewDirectory");
11. Console.WriteLine("NewDirectory is Created in C Directory");
12. Console.ReadLine();
14. }
15. }

advertisements

Here is the output of the C# Program:

NewDirectory is Created in C Directory

C# Program to View the Information of the File

This C# Program to View the Information of the File. Here the Attributes property that is used returns an enumerated constant that is encoded as enum flags.

Here is source code of the C# Program to View the Information of the File. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to View the Information of the File*
3. *\*/*
4. **using** System;
5. **using** System.IO;
6. **class** Program
7. {
8. **static** **void** Main()
9. {
10. FileInfo info = new FileInfo("C:**\\**sri**\\**srip.txt");
11. FileAttributes attributes = info.Attributes;
12. Console.WriteLine("Nature(Attribute) of the File : {0}",attributes);
13. Console.Read();
14. }
15. }

advertisements

Here is the output of the C# Program:

Nature(Attribute) of the File : Archive

C# Program to View the Date and time of Access of a File

This C# Program to View the Date and time of Access of a File. Here the date and time of the file access is obtained using the info class functions.

Here is source code of the C# Program to View the Date and time of Access of a File. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to View the Date and time of Access of a File*
3. *\*/*
4. **using** System;
5. **using** System.IO;
6. **class** Program
7. {
8. **static** **void** Main()
9. {
10. FileInfo info = new FileInfo("C:**\\**sri**\\**srip.txt");
11. DateTime time = info.CreationTime;
12. Console.WriteLine("File Creation Time : {0}", time);
13. time = info.LastAccessTime;
14. Console.WriteLine("File Last Access Time : {0}", time);
15. time = info.LastWriteTime;
16. Console.WriteLine("File Last Write Time : {0} ", time);
17. Console.Read();
18. }
19. }

Here is the output of the C# Program:

File Creation Time : 8/11/2013 7:17:20 PM

File Access Time : 8/15/2013 1:08:45 PM

File Last Write Time : 8/15/2013 1:37:36 PM

C# Program to Check the Existence of a File

This C# Program Checks the Existence of a File. Here the existence of the file is checked using the File.Exist function with a Boolean variable.

Here is source code of the C# Program to Check the Existence of a File. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Check the Existence of a File*
3. *\*/*
4. **using** System;
5. **using** System.IO;
6. **class** Program
7. {
8. **static** **void** Main()
9. {
10. FileInfo info = new FileInfo("C:**\\**sri**\\**srip.txt");
11. **bool** exists = info.Exists;
12. **if** (exists == **true**)
13. {
14. Console.WriteLine("The File Exists");
15. }
16. **else**
17. {
18. Console.WriteLine("No Such File Found");
19. }
20. Console.Read();
21. }
22. }

advertisements

Here is the output of the C# Program:

File Exists

#### 2. C# Examples demonstrating StringReader and StringBuilder

The StringReader class turns an ordinary string into a reader. The StringBuilder class is used to create mutable string. It is same as string class except that it is mutable. The programs in the section demonstrate StringReader and StringBuilder functions. The remaining programs in the section perform text operations in a file, search directories and list files, copies the contents from one file to another file and reads lines from a file until the end of file is reached.

C# Program to Perform Text Operations in a File

This C# Program Performs Text Operations in a File. Here appending text,creating a new text operations are performed.

Here is source code of the C# Program to Perform Text Operations in a File. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Perform Text Operations in a File*
3. *\*/*
4. **using** System;
5. **using** System.IO;
6. **class** Program
7. {
8. **static** **void** Main()
9. {
10. FileInfo finfo = new FileInfo("C:**\\**sri**\\**srip.txt");
11. **using** (StreamWriter writer = finfo.AppendText())
12. {
13. writer.WriteLine("New File with various Text operations");
14. }
15. finfo = new FileInfo("C:**\\**sri**\\**srip.txt");
16. **using** (StreamWriter writer = finfo.CreateText())
17. {
18. writer.WriteLine("New File with various Text operations");
19. }
20. **using** (StreamReader reader = finfo.OpenText())
21. {
22. Console.WriteLine(reader.ReadToEnd());
23. }
24. Console.Read();
25. }
26. }

advertisements

Here is the output of the C# Program:

New File with various Text operations

C# Program to Search Directories and List Files

This C# Program Searches Directories and List Files. Here the particular given directory is searched and all the files are listed.

Here is source code of the C# Program to Search Directories and List Files. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Search Directories and List Files*
3. *\*/*
4. **using** System;
5. **using** System.IO;
6. **class** Program
7. {
8. **static** **void** Main()
9. {
10. **string**[] Dirfile = Directory.GetFiles("C:**\\**sri**\\**","\*.\*",SearchOption.AllDirectories);
11. **foreach** (**string** file **in** Dirfile)
12. {
13. Console.WriteLine(file);
14. }
15. Console.Read();
16. }
17. }

advertisements

Here is the output of the C# Program:

The List of Files in the Directory are :

C:\sri\message.txt

C:\sri\srip.txt

C:\sri\test.txt

C# Program to Demonstrate StringReader

This C# Program Demonstrates StringReader. Here StringReader, the lines of a string are read individually in the order they appear. This type enables us to access string data through a stream-oriented interface.

Here is source code of the C# Program to Demonstrate StringReader. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Demonstrate StringReader*
3. *\*/*
4. **using** System;
5. **using** System.IO;
7. **class** Program
8. {
9. **const** **string** text = @"Sanfoundry
10. offers Training and Competency
11. development programs";
12. **static** **void** Main()
13. {
14. **using** (StringReader reader = new StringReader(text))
15. {
16. **int** count = 0;
17. **string** textline;
18. **while** ((textline = reader.ReadLine()) != **null**)
19. {
20. count++;
21. Console.WriteLine("Line {0}: {1}", count, textline);
22. }
23. Console.ReadLine();
24. }
25. }
26. }

advertisements

Here is the output of the C# Program:

Line 1 : Sanfoundry

Line 2 : Offers Training and Competency

Line 3 : development programs

C# Program to Illustrate StringBuilder

This C# Program Illustrates StringBuilder. Here ToString method is used on the StringBuilder type. This method is used to convert a StringBuilder’s character buffer into an actual string reference.

Here is source code of the C# Program to Illustrate StringBuilder. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Illustrate StringBuilder*
3. *\*/*
4. **using** System;
5. **using** System.Text;
6. **class** Program
7. {
8. **static** **void** Main()
9. {
10. StringBuilder bd = new StringBuilder();
11. bd.Append("1 ");
12. bd.Append("2 ");
13. bd.Append("3 ");
14. **for** (**int** i = 0; i < 5; i++)
15. {
16. bd.Append("z ");
17. }
18. **string** result = bd.ToString();
19. Console.WriteLine(result);
20. Console.ReadLine();
21. }
22. }

advertisements

Here is the output of the C# Program:

1 2 3 z z z z z

C# Program to Copy the Contents from one File to another File

This C# Program to Copy the Contents from one File to another File. Here File.Copy method is used to copy the contents of one file to another.

Here is source code of the C# Program to Copy the Contents from one File to another File. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Print the Sum of all the Multiples of 3 and 5*
3. *\*/*
4. **using** System;
5. **using** System.IO;
6. **class** Program
7. {
8. **static** **void** Main()
9. {
10. File.Copy("sri.txt", "srip.txt");
11. Console.WriteLine(File.ReadAllText("sri.txt"));
12. Console.WriteLine(File.ReadAllText("srip.txt"));
13. Console.Read();
14. }
15. }

advertisements

Here is the output of the C# Program:

Contents of File S

Contents of File S

C# Program to Read Lines from a File until the End of File is Reached

This C# Program Reads Lines from a File until the End of File is Reached. Here StreamWriter writes text files and this program read lines from a file that is created.

Here is source code of the C# Program to Read Lines from a File until the End of File is Reached. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Read Lines from a File until the End of File is Reached*
3. *\*/*
4. **using** System;
5. **using** System.IO;
6. **class** Test
7. {
8. **public** **static** **void** Main()
9. {
10. **string** path = @"c:\sri\srip.txt";
11. **try**
12. {
13. **if** (File.Exists(path))
14. {
15. File.Delete(path);
16. }
17. **using** (StreamWriter sw = new StreamWriter(path))
18. {
19. sw.WriteLine("This");
20. sw.WriteLine("text is");
21. sw.WriteLine("to test");
22. sw.WriteLine("Reading");
23. }
25. **using** (StreamReader sr = new StreamReader(path))
26. {
27. **while** (sr.Peek() >= 0)
28. {
29. Console.WriteLine(sr.ReadLine());
30. }
31. }
32. }
33. **catch** (Exception e)
34. {
35. Console.WriteLine("The process failed: {0}", e.ToString());
36. }
37. Console.Read();
38. }
39. }

advertisements

Here is the output of the C# Program:

This

text is

to test

reading

#### 3. C# Examples demonstrating StringWriter, StreamReader and Memory Stream

StringWriter writes to a string synchronously or asynchronously. It is found in the System.IO namespace. The FileInfo class provides instance methods for the creation, copying, deletion, moving, and opening of files. The MemoryStream creates a stream whose backing store is memory. The programs in the following section lists Disk Drives, Illustrates StringWriter, uses StreamReader to read entire line, get file time using file class, read data from stream and cast data to chars, illustrates memory stream class and methods of FileInfo Class.

C# Program to List Disk Drives

This C# Program Lists Disk Drives. Here the disk drives are listed using DriveInfo.GetDrives function.

Here is source code of the C# Program to List Disk Drives. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to List Disk Drives*
3. *\*/*
4. **using** System;
5. **using** System.IO;
6. **class** Test
7. {
8. **public** **static** **void** Main()
9. {
10. DriveInfo[] driverslist = DriveInfo.GetDrives();
11. **foreach** (DriveInfo d **in** driverslist)
12. {
13. Console.WriteLine("Drive {0}", d.Name);
14. Console.WriteLine(" File type: {0}", d.DriveType);
15. **if** (d.IsReady == **true**)
16. {
17. Console.WriteLine(" Total size of drive:{0, 15} bytes ",d.TotalSize);
18. Console.Read();
19. }
20. }
21. }
22. }

advertisements

Here is the output of the C# Program:

Drive C:\

File Type : Fixed

Total Size of Drive : 107268272128

C# Program to Illustrate StringWriter

This C# Program Illustrates StringWriter. Here it implements a TextWriter for writing information to a string. The information is stored in an underlying StringBuilder..

Here is source code of the C# Program to Illustrate StringWriter. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Illustrate StringWriter*
3. *\*/*
4. **using** System;
5. **using** System.IO;
6. **using** System.Text;
7. **public** **class** stringwrt
8. {
9. StringBuilder sb = new StringBuilder();
10. **public** stringwrt()
11. {
12. Writer();
13. }
14. **public** **static** **void** Main()
15. {
16. stringwrt srw = new stringwrt();
17. }
18. **private** **void** Writer()
19. {
20. StringWriter sw = new StringWriter(sb);
21. Console.WriteLine("STUDENT DETAILS : ");
22. Console.Write("Name :");
23. **string** name = Console.ReadLine();
24. sw.WriteLine("Name :" + name);
25. Console.Write("Department :");
26. **string** Department = Console.ReadLine();
27. sw.WriteLine("Department :" + Department);
28. Console.Write("College Name :");
29. **string** CollegeName = Console.ReadLine();
30. sw.WriteLine("College Name :" + CollegeName);
31. Console.WriteLine("Information Saved!");
32. Console.WriteLine();
33. sw.Flush();
34. sw.Close();
35. Console.ReadLine();
36. }
37. }

advertisements

Here is the output of the C# Program:

STUDENT DETAILS :

Name : BOB

Department : IT

College Name : NIIT

Information Saved!

C# Program to Use StreamReader to Read Entire Line

This C# Program to Use StreamReader to Read Entire Line. Here with the help of streamreader it reads and displays the entire line from the file.

Here is source code of the C# Program to Use StreamReader to Read Entire Line. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Use StreamReader to Read Entire Line*
3. *\*/*
4. **using** System;
5. **using** System.IO;
6. **using** System.IO.Compression;
7. **using** System.Text;
8. **public** **sealed** **class** Program
9. {
10. **public** **static** **void** Main()
11. {
12. Stream s = new FileStream(@"c:\sri\srip.txt", FileMode.Open);
13. **using** (StreamReader sr = new StreamReader(s, Encoding.UTF8))
14. {
15. **string** line;
16. **while** ((line = sr.ReadLine()) != **null**)
17. {
18. Console.WriteLine(line);
19. }
20. Console.ReadLine();
21. }
22. }
23. }

advertisements

Here is the output of the C# Program:

StreamWriter writes text files. It enables easy and efficient text output. It is best placed in a using-statement to ensure it is removed from memory when no longer needed. It provides several constructors and many methods.

C# Program to Get File Time using File Class

This C# Program Gets File Time using File Class. Here the File creation time is found using the date time class.

Here is source code of the C# Program to Get File Time using File Class. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Get File Time using File Class*
3. *\*/*
4. **using** System;
5. **using** System.IO;
7. **class** Program
8. {
9. **static** **void** Main()
10. {
11. FileInfo info = new FileInfo("C:**\\**srip.txt");
12. DateTime time = info.CreationTime;
13. Console.WriteLine("File was Created at : ");
14. Console.Write(time);
15. Console.Read();
16. }
17. }

advertisements

Here is the output of the C# Program:

File was Created at :

9/30/2013 12:15:44 PM

C# Program to Read Data from Stream and Cast Data to Chars

This C# Program Reads Data from Stream and Cast Data to Chars. Here the string from the file is converted into character data.

Here is source code of the C# Program to Read Data from Stream and Cast Data to Chars. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Read Data from Stream and Cast Data to Chars*
3. *\*/*
4. **using** System;
5. **using** System.IO;
6. **public** **sealed** **class** Program
7. {
8. **public** **static** **void** Main()
9. {
10. **using** (Stream s = new FileStream(@"c:\sri\srip.txt", FileMode.Open))
11. {
12. **int** read;
13. **while** ((read = s.ReadByte()) != -1)
14. {
15. Console.Write("{0} ", (**char**)read);
16. }
17. Console.ReadLine();
18. }
19. }
20. }

advertisements

Here is the output of the C# Program:

G O O D M O R N I N G

C# Program to Illustrate Memory Stream Class

This C# Program Illustrates Memory Stream Class. Here the below program shows how to read and write data using memory as a backing store.

Here is source code of the C# Program to Illustrate Memory Stream Class. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Illustrate Memory Stream Class*
3. *\*/*
4. **using** System;
5. **using** System.IO;
6. **using** System.Text;
7. **class** MemStream
8. {
9. **static** **void** Main()
10. {
11. **int** count;
12. **byte**[] byteArray;
13. **char**[] charArray;
14. UnicodeEncoding uniEncoding = new UnicodeEncoding();
15. **byte**[] firstString = uniEncoding.GetBytes("Invalid file path characters are: ");
16. **byte**[] secondString = uniEncoding.GetBytes(Path.GetInvalidPathChars());
17. **using**(MemoryStream memStream = new MemoryStream(100))
18. {
19. memStream.Write(firstString, 0 , firstString.Length);
20. count = 0;
21. **while**(count < secondString.Length)
22. {
23. memStream.WriteByte(secondString[count++]);
24. }
25. Console.WriteLine("Capacity = {0}, Length = {1}, Position = {2}**\n**",
26. memStream.Capacity.ToString(),
27. memStream.Length.ToString(),
28. memStream.Position.ToString());
29. memStream.Seek(0, SeekOrigin.Begin);
30. byteArray = new **byte**[memStream.Length];
31. count = memStream.Read(byteArray, 0, 20);
32. **while** (count < memStream.Length)
33. {
34. byteArray[count++] = Convert.ToByte(memStream.ReadByte());
35. }
36. charArray = new **char**[uniEncoding.GetCharCount(byteArray, 0, count)];
37. uniEncoding.GetDecoder().GetChars(byteArray, 0, count, charArray, 0);
38. Console.WriteLine(charArray);
39. Console.Read();
40. }
41. }
42. }

advertisements

Here is the output of the C# Program:

Capacity = 256 Length = 140 Position =140

Invalid File Path Characters are : "<>|

C# Program to Illustrate Methods of FileInfo Class

This C# Program Illustrates Methods of FileInfo Class. Here Use the FileInfo class for typical operations such as copying, moving, renaming, creating, opening, deleting, and appending to files.

Here is source code of the C# Program to Illustrate Methods of FileInfo Class. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Illustrate Methods of FileInfo Class*
3. *\*/*
4. **using** System;
5. **using** System.IO;
6. **class** Test
7. {
8. **public** **static** **void** Main()
9. {
10. **string** path = Path.GetTempFileName();
11. FileInfo fi1 = new FileInfo(path);
12. **using** (StreamWriter sw = fi1.CreateText())
13. {
14. sw.WriteLine("This is");
15. sw.WriteLine("Sanfoundry");
16. sw.WriteLine("Website");
17. }
18. **using** (StreamReader sr = fi1.OpenText())
19. {
20. **string** s = "";
21. **while** ((s = sr.ReadLine()) != **null**)
22. {
23. Console.WriteLine(s);
24. }
25. }
26. **try**
27. {
28. **string** path2 = Path.GetTempFileName();
29. FileInfo fi2 = new FileInfo(path2);
30. fi2.Delete();
31. fi1.CopyTo(path2);
32. Console.WriteLine("{0} was copied to {1}.", path, path2);
33. fi2.Delete();
34. Console.WriteLine("{0} was successfully deleted.", path2);
35. }
36. **catch** (Exception e)
37. {
38. Console.WriteLine("The process failed: {0}", e.ToString());
39. }
40. Console.Read();
41. }
42. }

advertisements

Here is the output of the C# Program:

This is

Sanfoundry

Website.

C:\Users\win7\AppData\Local\Temp\tmpAEF8.tmp was copied to C:\users\Win7\AppData\Local\Temp\tmpAEF7.tmp

C:\users\Win7\AppData\Local\Temp\tmpAEF8.tmp was successfully deleted.

#### 4. C# Examples implementing Binary Reader and other File Functions

Binary Reader reads the number of bytes from the current stream into a array and advances the current position by number of bytes that are read. The following section contains programs which get content from a file and reads the content 1 byte at a time, lists the files in a directory, calculates the size of a Folder, implements BinaryReader and performs file comparison.

C# Program to Get Content from a File and Read the Content 1 Byte at a Time

This C# Program Gets Content from a File and Read the Content 1 Byte at a Time. Here it firsts gets the contents from a file and reads the content per byte.

Here is source code of the C# Program to Get Content from a File and Read the Content 1 Byte at a Time. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Get Content from a File and Read the Content 1 Byte at a Time*
3. *\*/*
4. **using** System;
5. **using** System.IO;
6. **public** **sealed** **class** Program
7. {
8. **public** **static** **void** Main()
9. {
10. **using** (Stream s = new FileStream(@"c:\sri\srip.txt", FileMode.Open))
11. {
12. **int** read;
13. **while** ((read = s.ReadByte()) != -1)
14. {
15. Console.Write("{0} ", read);
16. }
17. Console.ReadLine();
18. }
19. }
20. }

advertisements

Here is the output of the C# Program:

71 79 79 68 77 79 82 78 73 78 71

C# Program to List the Files in a Directory

This C# Program Lists the Files in a Directory. Here files in the directory are displayed using directory.getfiles().

Here is source code of the C# Program to List the Files in a Directory. The C# program is ,successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to List the Files in a Directory*
3. *\*/*
4. **using** System;
5. **using** System.IO;
6. **class** Program
7. {
8. **static** **void** Main()
9. {
10. **string**[] array1 = Directory.GetFiles(@"D:\");
11. Console.WriteLine("Files in the Directory");
12. **foreach** (**string** name **in** array1)
13. {
14. Console.WriteLine(name);
15. }
16. Console.Read();
17. }
18. }

advertisements

Here is the output of the C# Program:

Files in the Directory

D:\demo1.cs

D:\demo1.exe

D:\msdia80.dll

D:\demo1.txt

C# Program to Trap Events from File

This C# Program to Trap Events from File. Here the events that happen in the specified locations are trapped.

Here is source code of the C# Program to Trap Events from File. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Trap Events from File*
3. *\*/*
4. **using** System;
5. **using** System.IO;
6. **class** Test
7. {
8. **static** **void** namechang(**object** sender, RenamedEventArgs evn)
9. {
10. Console.WriteLine("{0} NameChanged to {1}", evn.OldFullPath, evn.FullPath);
11. }
12. **static** **void** changed(**object** sender, FileSystemEventArgs evn)
13. {
14. Console.WriteLine(evn.FullPath + " " + evn.ChangeType);
15. }
16. **static** **void** Main(**string**[] arg)
17. {
18. FileSystemWatcher w = new FileSystemWatcher();
19. w.Path = "d:**\\**srip";
20. w.NotifyFilter = NotifyFilters.FileName | NotifyFilters.DirectoryName |NotifyFilters.LastAccess | NotifyFilters.LastWrite;
21. w.Filter = "";
22. w.Created += new FileSystemEventHandler(changed);
23. w.Deleted += new FileSystemEventHandler(changed);
24. w.Changed += new FileSystemEventHandler(changed);
25. w.Renamed += new RenamedEventHandler(namechang);
26. w.EnableRaisingEvents = **true**;
27. Console.WriteLine("Press any key to quit");
28. Console.Read();
29. }
30. }

advertisements

Here is the output of the C# Program:

Press any key to quit

C# Program to Calculate the Size of Folder

This C# Program Calculates the Size of Folder. Here the program calculates the size of the folder including its subfolders and hence display the size in bytes,kilobytes and MB.

Here is source code of the C# Program to Calculate the Size of Folder. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Calculate the Size of Folder*
3. *\*/*
4. **using** System;
5. **using** System.Linq;
6. **using** System.IO;
7. **namespace** ConsoleApplication3
8. {
9. **class** Program
10. {
11. **static** **void** Main(**string**[] args)
12. {
13. DirectoryInfo dInfo = new DirectoryInfo(@"C:/sri");
14. **long** sizeOfDir = DirectorySize(dInfo, **true**);
15. Console.WriteLine("Directory size in Bytes : " +
16. "{0:N0} Bytes", sizeOfDir);
17. Console.WriteLine("Directory size in KB : " +
18. "{0:N2} KB", ((**double**)sizeOfDir) / 1024);
19. Console.WriteLine("Directory size in MB : " +
20. "{0:N2} MB", ((**double**)sizeOfDir) / (1024 \* 1024));
21. Console.ReadLine();
22. }
23. **static** **long** DirectorySize(DirectoryInfo dInfo, **bool** includeSubDir)
24. {
25. **long** totalSize = dInfo.EnumerateFiles()
26. .Sum(file => file.Length);
27. **if** (includeSubDir)
28. {
29. totalSize += dInfo.EnumerateDirectories()
30. .Sum(dir => DirectorySize(dir, **true**));
31. }
32. **return** totalSize;
33. }
34. }
35. }

advertisements

Here is the output of the C# Program:

Directory Size in Bytes : 1,482 Bytes

Directory Size in KB : 1.45 KB

Directory Size in MB : 0.00 MB

C# Program to Implement BinaryReader

This C# Program Implements BinaryReader . Here the it Reads primitive data types as binary values in a specific encoding..

Here is source code of the C# Program to Implement BinaryReader . The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Implement BinaryReader*
3. *\*/*
4. **using** System;
5. **using** System.IO;
6. **class** ConsoleApplication
7. {
8. **const** **string** fileName = "program.dat";
9. **static** **void** Main()
10. {
11. Write();
12. Console.WriteLine("Using Binary Writer Class the Contents are Written ");
13. Display();
14. }
15. **public** **static** **void** Write()
16. {
17. **using** (BinaryWriter writer = new BinaryWriter(File.Open(fileName, FileMode.Create)))
18. {
19. writer.Write(1.250F);
20. writer.Write(@"C:\Temp");
21. }
22. }
23. **public** **static** **void** Display()
24. {
25. **float** aspectRatio;
26. **string** tempDirectory;
27. **if** (File.Exists(fileName))
28. {
29. **using** (BinaryReader reader = new BinaryReader(File.Open(fileName, FileMode.Open)))
30. {
31. aspectRatio = reader.ReadSingle();
32. tempDirectory = reader.ReadString();
33. }
34. Console.WriteLine("Aspect Ratio Set to : " + aspectRatio);
35. Console.WriteLine("Temp Directory is : " + tempDirectory);
36. Console.Read();
37. }
38. }
39. }

advertisements

Here is the output of the C# Program:

Using Binary Writer Class the Contents are Written

Aspect Ratio set to : 1.25

Temp Directory is : C:\Temp

C# Program to Perform File Comparison

This C# Program Performs File Comparison. Here the files are compared and based on the equality the results are displayed.

Here is source code of the C# Program to Perform File Comparison. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Perform File Comparison*
3. *\*/*
4. **using** System;
5. **using** System.Threading;
6. **using** System.IO;
8. **class** Reader
9. {
10. **string** fileName;
11. **public** **string** data;
13. **public** Reader(**string** fn) { fileName = fn; }
15. **public** **void** Read()
16. {
17. FileStream s = new FileStream(fileName, FileMode.Open);
18. StreamReader r = new StreamReader(s);
19. data = r.ReadToEnd();
20. r.Close();
21. s.Close();
23. }
24. }
25. **class** ThreadSample
26. {
27. **static** **void** Main(**string**[] arg)
28. {
29. **if** (arg.Length == 2)
30. {
31. Reader a = new Reader(arg[0]);
32. Reader b = new Reader(arg[1]);
33. Thread ta = new Thread(new ThreadStart(a.Read));
34. Thread tb = new Thread(new ThreadStart(b.Read));
35. ta.Start();
36. tb.Start();
37. ta.**Join**();
38. tb.**Join**();
40. **if** (a.data.Length == b.data.Length)
41. {
42. **int** i = 0;
43. **while** (i < a.data.Length && a.data[i] == b.data[i]) i++;
44. **if** (i == a.data.Length)
45. Console.WriteLine("Files {0} and {1} are equal", arg[0], arg[1]);
46. **else**
47. Console.WriteLine("Files {0} and {1} are not equal", arg[0], arg[1]);
48. }
49. **else**
50. {
51. Console.WriteLine("Files {0} and {1} are not equal", arg[0], arg[1]);
52. }
53. }
54. **else**
55. {
56. Console.WriteLine("-- enter two file names");
57. }
58. Console.ReadLine();
59. }
60. }

advertisements

Here is the output of the C# Program:

D:\Desktop\c#\program codes>pgno382.exe d:\\sri\\File1.txt d:\\sri\\File1.txt

Files d:\\sri\\File1.txt and d:\\sri\\File1.txt

are equal

#### M. C# Programming Examples on DataStructures

The programs in this section illustrate the different types of data structures.

#### C# Examples on Different Types of Data Structures

A Linked List is a dynamic data structure which is made up of a sequence of nodes. A Binary Search Tree (BST) is a tree in which each vertex can hold upto 2 children. The left child must hold a value smaller than its own, and the right child must hold a value larger than its own. Stack is an abstract data type which represents a collection of elements. The two operations that can be performed on a stack are Push and Pop. Push operation keeps adding new elements into a stack. Pop operation removes the last element that was added onto a stack. The following programs implement a Binary Search Tree using Linked List, creates a Singly Linked Circular List, implements traversal in a Singly Linked List, implements a Stack with Push and Pop operations and creates an instance of StackTrace to get all frames.

C# Program to Implement Binary Search Tree using Linked List

This C# Program Implements Binary Search Tree using Linked List.A Binary tree is a tree data structure in which each node has at most two child nodes, usually distinguished as “left” and “right”.

Here is source code of the C# Program to Implement Binary Search Tree using Linked List. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Implement Binary Search Tree using Linked List*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Text;
7. **namespace** TreeSort
8. {
9. **class** Node
10. {
11. **public** **int** item;
12. **public** Node leftc;
13. **public** Node rightc;
14. **public** **void** display()
15. {
16. Console.Write("[");
17. Console.Write(item);
18. Console.Write("]");
19. }
20. }
21. **class** Tree
22. {
23. **public** Node root;
24. **public** Tree()
25. {
26. root = **null**;
27. }
28. **public** Node ReturnRoot()
29. {
30. **return** root;
31. }
32. **public** **void** Insert(**int** id)
33. {
34. Node newNode = new Node();
35. newNode.item = id;
36. **if** (root == **null**)
37. root = newNode;
38. **else**
39. {
40. Node current = root;
41. Node parent;
42. **while** (**true**)
43. {
44. parent = current;
45. **if** (id < current.item)
46. {
47. current = current.leftc;
48. **if** (current == **null**)
49. {
50. parent.leftc = newNode;
51. **return**;
52. }
53. }
54. **else**
55. {
56. current = current.rightc;
57. **if** (current == **null**)
58. {
59. parent.rightc = newNode;
60. **return**;
61. }
62. }
63. }
64. }
65. }
66. **public** **void** Preorder(Node Root)
67. {
68. **if** (Root != **null**)
69. {
70. Console.Write(Root.item + " ");
71. Preorder(Root.leftc);
72. Preorder(Root.rightc);
73. }
74. }
75. **public** **void** Inorder(Node Root)
76. {
77. **if** (Root != **null**)
78. {
79. Inorder(Root.leftc);
80. Console.Write(Root.item + " ");
81. Inorder(Root.rightc);
82. }
83. }
84. **public** **void** Postorder(Node Root)
85. {
86. **if** (Root != **null**)
87. {
88. Postorder(Root.leftc);
89. Postorder(Root.rightc);
90. Console.Write(Root.item + " ");
91. }
92. }
93. }
94. **class** Program
95. {
96. **static** **void** Main(**string**[] args)
97. {
98. Tree theTree = new Tree();
99. theTree.Insert(20);
100. theTree.Insert(25);
101. theTree.Insert(45);
102. theTree.Insert(15);
103. theTree.Insert(67);
104. theTree.Insert(43);
105. theTree.Insert(80);
106. theTree.Insert(33);
107. theTree.Insert(67);
108. theTree.Insert(99);
109. theTree.Insert(91);
110. Console.WriteLine("Inorder Traversal : ");
111. theTree.Inorder(theTree.ReturnRoot());
112. Console.WriteLine(" ");
113. Console.WriteLine();
114. Console.WriteLine("Preorder Traversal : ");
115. theTree.Preorder(theTree.ReturnRoot());
116. Console.WriteLine(" ");
117. Console.WriteLine();
118. Console.WriteLine("Postorder Traversal : ");
119. theTree.Postorder(theTree.ReturnRoot());
120. Console.WriteLine(" ");
121. Console.ReadLine();
122. }
123. }
124. }

advertisements

Here is the output of the C# Program:

Inorder Traversal :

15 20 25 33 43 45 67 67 80 91 99

Preorder Traversal :

20 15 25 45 43 33 67 80 67 99 91

Postorder Traversal :

15 33 43 67 91 99 80 67 45 25 20

C# Program to Create an Instance of StackTrace and to Get all Frames

This C# Program Creates an Instance of StackTrace and to Get all Frames. Here It creates an instance of StackTrace (call stack), gets all frames (method calls) and writes the method names.

Here is source code of the C# Program to Create an Instance of StackTrace and to Get all Frames. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Create an Instance of StackTrace and to Get all Frames*
3. *\*/*
4. **using** System.Diagnostics;
5. **using** System;
6. **class** program
7. {
8. **public** **static** **void** Main()
9. {
10. StackTrace stackTrace = new StackTrace();
11. StackFrame[] stackFrames = stackTrace.GetFrames();
12. *// write call stack method names*
13. Console.WriteLine("Method Names : ");
14. **foreach** (StackFrame stackFrame **in** stackFrames)
15. {
16. Console.WriteLine(stackFrame.GetMethod().Name);
17. }
18. Console.Read();
19. }
20. }

advertisements

Here is the output of the C# Program:

Method Names :

Main

nExecuteAssembly

ExecuteAssembly

RunUsersAssembly

ThreadStart\_Context

Run

ThreadStart

C# Program to Create a Singly Linked Circular List

This C# Program Creates a Singly Linked Circular List. Here singly linked circular list, filling elements and traversal in forward direction and counting the number of elements in the list is done.

Here is source code of the C# Program to Create a Singly Linked Circular List. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Create a Singly Linked Circular List*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
8. **namespace** CSTest
9. {
10. **class** Circlist
11. {
12. **private** **int** currentdata;
13. **private** Circlist nextdata;
14. **public** Circlist()
15. {
16. currentdata = 0;
17. nextdata = **this**;
18. }
19. **public** Circlist(**int** **value**)
20. {
21. currentdata = **value**;
22. nextdata = **this**;
23. }
24. **public** Circlist Insdata(**int** **value**)
25. {
26. Circlist node = new Circlist(**value**);
27. **if** (**this**.nextdata == **this**)
28. {
29. node.nextdata = **this**;
30. **this**.nextdata = node;
31. }
32. **else**
33. {
34. Circlist temp = **this**.nextdata;
35. node.nextdata = temp;
36. **this**.nextdata = node;
37. }
38. **return** node;
39. }
40. **public** **int** Deldata()
41. {
42. **if** (**this**.nextdata == **this**)
43. {
44. System.Console.WriteLine("**\n**Only one node!!!!");
45. **return** 0;
46. }
47. Circlist node = **this**.nextdata;
48. **this**.nextdata = **this**.nextdata.nextdata;
49. node = **null**;
50. **return** 1;
51. }
52. **public** **void** Traverse()
53. {
54. Traverse(**this**);
55. }
56. **public** **void** Traverse(Circlist node)
57. {
58. **if** (node == **null**)
59. node = **this**;
60. System.Console.WriteLine("Forward Direction!!!!");
61. Circlist snode = node;
62. **do**
63. {
64. System.Console.WriteLine(node.currentdata);
65. node = node.nextdata;
66. }
67. **while** (node != snode);
68. }
69. **public** **int** Gnodes()
70. {
71. **return** Gnodes(**this**);
72. }
73. **public** **int** Gnodes(Circlist node)
74. {
75. **if** (node == **null**)
76. node = **this**;
77. **int** count = 0;
78. Circlist snode = node;
79. **do**
80. {
81. count++;
82. node = node.nextdata;
83. }
84. **while** (node != snode);
85. System.Console.WriteLine("**\n**Current Node Value : " + node.currentdata.ToString());
86. System.Console.WriteLine("**\n**Total nodes :" + count.ToString());
87. **return** count;
88. }
89. **static** **void** Main(**string**[] args)
90. {
91. Circlist node1 = new Circlist(1);
92. node1.Deldata();
93. Circlist node2 = node1.Insdata(2);
94. node1.Deldata();
95. node2 = node1.Insdata(2);
96. Circlist node3 = node2.Insdata(3);
97. Circlist node4 = node3.Insdata(4);
98. Circlist node5 = node4.Insdata(5);
99. node1.Gnodes();
100. node3.Gnodes();
101. node5.Gnodes();
102. node1.Traverse();
103. node3.Deldata();
104. node2.Traverse();
105. node1.Gnodes();
106. node3.Gnodes();
107. node5.Gnodes();
108. Console.Read();
109. }
110. }
111. }

advertisements

Here is the output of the C# Program:

Only one Node!!!!!

Current Node Value : 1

Total nodes : 5

Current Node Value : 3

Total nodes : 5

Current Node Value : 5

Total nodes : 5

Forward Direction!!!!

1

2

3

4

5

Forward Direction!!!!

2

3

5

1

Current Node Value : 1

Total nodes : 4

Current Node Value : 3

Total nodes : 4

Current Node Value: 5

Total nodes : 4

C# Program to Implement Stack with Push and Pop operations

This C# Program Implements Stack with Push and Pop operations. Here Push enters an item on the stack, and pop retrieves an item, moving the rest of the items in the stack up one level.

Here is source code of the C# Program to Implement Stack with Push and Pop operations. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Implement Stack with Push and Pop operations*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
8. **using** System.Threading.Tasks;
9. **namespace** ConsoleApplication1
10. {
11. **class** Program
12. {
13. **static** **void** Main(**string**[] args)
14. {
15. stack st = new stack();
16. **while** (**true**)
17. {
18. Console.Clear();
19. Console.WriteLine("**\n**Stack MENU(size -- 10)");
20. Console.WriteLine("1. Add an element");
21. Console.WriteLine("2. See the Top element.");
22. Console.WriteLine("3. Remove top element.");
23. Console.WriteLine("4. Display stack elements.");
24. Console.WriteLine("5. Exit");
25. Console.Write("Select your choice: ");
26. **int** choice = Convert.ToInt32(Console.ReadLine());
27. **switch** (choice)
28. {
29. **case** 1:
30. Console.WriteLine("Enter an Element : ");
31. st.Push(Console.ReadLine());
32. **break**;
34. **case** 2: Console.WriteLine("Top element is: {0}", st.Peek());
35. **break**;
37. **case** 3: Console.WriteLine("Element removed: {0}", st.Pop());
38. **break**;
40. **case** 4: st.Display();
41. **break**;
43. **case** 5: System.Environment.Exit(1);
44. **break**;
45. }
46. Console.ReadKey();
47. }
48. }
49. }
51. **interface** StackADT
52. {
53. Boolean isEmpty();
54. **void** Push(**Object** element);
55. **Object** Pop();
56. **Object** Peek();
57. **void** Display();
58. }
59. **class** stack : StackADT
60. {
61. **private** **int** StackSize;
62. **public** **int** StackSizeSet
63. {
64. **get** { **return** StackSize; }
65. **set** { StackSize = **value**; }
66. }
67. **public** **int** top;
68. **Object**[] item;
69. **public** stack()
70. {
71. StackSizeSet = 10;
72. item = new **Object**[StackSizeSet];
73. top = -1;
74. }
75. **public** stack(**int** capacity)
76. {
77. StackSizeSet = capacity;
78. item = new **Object**[StackSizeSet];
79. top = -1;
80. }
81. **public** **bool** isEmpty()
82. {
83. **if** (top == -1) **return** **true**;
85. **return** **false**;
86. }
87. **public** **void** Push(**object** element)
88. {
89. **if** (top == (StackSize - 1))
90. {
91. Console.WriteLine("Stack is full!");
92. }
94. **else**
95. {
97. item[++top] = element;
98. Console.WriteLine("Item pushed successfully!");
99. }
100. }
101. **public** **object** Pop()
102. {
103. **if** (isEmpty())
104. {
105. Console.WriteLine("Stack is empty!");
106. **return** "No elements";
107. }
108. **else**
109. {
111. **return** item[top--];
112. }
113. }
114. **public** **object** Peek()
115. {
116. **if** (isEmpty())
117. {
119. Console.WriteLine("Stack is empty!");
120. **return** "No elements";
121. }
122. **else**
123. {
124. **return** item[top];
125. }
126. }

129. **public** **void** Display()
130. {
131. **for** (**int** i = top; i > -1; i--)
132. {
134. Console.WriteLine("Item {0}: {1}", (i + 1), item[i]);
135. }
136. }
137. }
138. }

advertisements

Here is the output of the C# Program:

Stack MENU(size -- 10)

1. Add an element

2. See the Top Element

3. Remove the Top Element

4. Display Stack Elements

5. Exit

Select your Choice : 1

Enter the Element : 25

Item Pushed Successfully!

Select your choice :1

Enter the Element : 26

Item Pushed Successfully!

Select your choice : 4

Item 2 :26

Item 1 :25

C# Program to Implement Traversal in Singly Linked List

This C# Program Implements Traversal in Singly Linked List. Here Elements are added in the singly linked list and the traversal in done in the forward direction.

Here is source code of the C# Program to Implement Traversal in Singly Linked List. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Implement Traversal in Singly Linked List*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
8. **namespace** Program
9. {
10. **class** singlelist
11. {
12. **private** **int** data;
13. **private** singlelist next;
14. **public** singlelist()
15. {
16. data = 0;
17. next = **null**;
18. }
19. **public** singlelist(**int** **value**)
20. {
21. data = **value**;
22. next = **null**;
23. }
24. **public** singlelist InsertNext(**int** **value**)
25. {
27. singlelist node = new singlelist(**value**);
28. **if** (**this**.next == **null**)
29. {
30. node.next = **null**;
31. **this**.next = node;
32. }
33. **else**
34. {
35. singlelist temp = **this**.next;
36. node.next = temp;
37. **this**.next = node;
38. }
39. **return** node;
40. }
41. **public** **int** DeleteNext()
42. {
43. **if** (next == **null**)
44. **return** 0;
45. singlelist node = **this**.next;
46. **this**.next = **this**.next.next;
47. node = **null**;
48. **return** 1;
49. }
50. **public** **void** Traverse(singlelist node)
51. {
52. **if** (node == **null**)
53. node = **this**;
54. System.Console.WriteLine("Traversing :");
55. **while** (node != **null**)
56. {
57. System.Console.WriteLine(node.data);
58. node = node.next;
59. }
60. }
61. }
62. **class** Program
63. {
64. **static** **void** Main(**string**[] args)
65. {
66. singlelist node1 = new singlelist(11);
67. singlelist node2 = node1.InsertNext(12);
68. singlelist node3 = node2.InsertNext(13);
69. singlelist node4 = node3.InsertNext(14);
70. singlelist node5 = node4.InsertNext(15);
71. node1.Traverse(**null**);
72. Console.WriteLine("Deleting !!");
73. node3.DeleteNext();
74. node2.Traverse(**null**);
75. System.Console.ReadLine();
76. }
77. }
78. }

advertisements

Here is the output of the C# Program:

Traversing :

11

12

13

14

15

Deleting!!!

Traversing :

12

13

14

#### N. C# Programming Examples on Mathematics

The programs in this section demonstrate various concepts of mathematics. These include special numbers, complex numbers, trigonometric functions, summation functions, quadratic equations, permutations and combinations, volume and surface area calculation, bitwise operations and various divisibility tests.

#### 1. C# Examples on Special Numbers in Mathematics

The C# Programs in the following section checks for different number types. These numbers include Armstrong number, Perfect number and Amicable number. It also generates various number series including Fibonacci series and Factorial series. It also performs summation operation on the given set of input numbers and performs arithmetic operations like Addition, Subtraction, Multiplication and Division on the given set of input numbers.

C# Program to Generate Fibonacci Series

This C# Program generates Fibonacci series.The numbers that precedes the series are 0 and 1.The next number is found by adding up the two numbers before it.

Here is source code of the C# program which generates a Fibonacci series.The C# program is successfully compiled and executed with Microsoft Visual Studio.The program output is also shown below.

1. */\**
2. *\* C# Program to Generate Fibonacci Series*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
9. **namespace** fibonaci
10. {
11. **class** Program
12. {
13. **static** **void** Main(**string**[] args)
14. {
15. **int** i, count, f1 = 0, f2 = 1, f3 = 0;
16. Console.Write("Enter the Limit : ");
17. count = **int**.Parse(Console.ReadLine());
18. Console.WriteLine(f1);
19. Console.WriteLine(f2);
20. **for** (i = 0; i <= count; i++)
21. {
22. f3 = f1 + f2;
23. Console.WriteLine(f3);
24. f1 = f2;
25. f2 = f3;
26. }
27. Console.ReadLine();
29. }
30. }
31. }

advertisements

Here is the output of the C# Program:

Enter the Limit : 10

0

1

1

2

3

5

8

13

21

34

55

89

144

C# Program to Generate the Factorial of Given Number

This C# Program generates Factorial of the Number obtained from the user. Factorial of a number is obtained from the result of multiplying a series of descending natural numbers.

Here is source code of the C# Program to Generate the Factorial of Given Number. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Generate the Factorial of Given Number*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
9. **namespace** factorial
10. {
11. **class** Program
12. {
13. **static** **void** Main(**string**[] args)
14. {
15. **int** i, number, fact;
16. Console.WriteLine("Enter the Number");
17. number = **int**.Parse(Console.ReadLine());
18. fact = number;
19. **for** (i = number - 1; i >= 1; i--)
20. {
21. fact = fact \* i;
22. }
23. Console.WriteLine("**\n**Factorial of Given Number is: "+fact);
24. Console.ReadLine();
26. }
27. }
28. }

advertisements

Here is the output of the C# Program:

Enter the Number

6

Factorial of Given Number is: 720

C# Program to Generate the Sum of N Numbers

This C# Program Generates the Sum of N Numbers.This C# program obtains the Nth number from the user and calculates its sum till the Nth number.

Here is source code of the C# Program to Generate the Sum of N Numbers. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Generate the Sum of N Numbers*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
9. **namespace** program
10. {
11. **class** Program
12. {
13. **static** **void** Main(**string**[] args)
14. {
15. **int** i, sum = 0,n;
16. Console.Write("Enter the Nth Number : ");
17. n = **int**.Parse(Console.ReadLine());
18. **for** (i = 0; i <= n; i++)
19. {
20. sum = sum + i;
21. }
22. Console.WriteLine("**\n**Sum of N Numbers : " + sum);
23. Console.ReadLine();
25. }
26. }
27. }

advertisements

Here is the output of the C# Program:

Enter the Nth Number : 10

Sum of N Numbers : 55

C# Program to Check Whether the Entered Number is an Armstrong Number or Not

This C# Program is written to Check Whether the Entered Number is an Armstrong Number or Not . An Armstrong number of three digits is an integer such that the sum of the cubes of its digits is equal to the number itself.

Here is source code of the C# Program to Check Whether the Entered Number is an Armstrong Number or Not. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Check Whether the Entered Number is an Armstrong Number or Not*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
9. **namespace** ConsoleApplication6
10. {
11. **class** Program
12. {
13. **static** **void** Main(**string**[] args)
14. {
15. **int** number, remainder, sum = 0;
16. Console.Write("enter the Number");
17. number = **int**.Parse(Console.ReadLine());
18. **for** (**int** i = number; i > 0; i = i / 10)
19. {
20. remainder = i % 10;
21. sum = sum + remainder\*remainder\*remainder;
23. }
24. **if** (sum == number)
25. {
26. Console.Write("Entered Number is an Armstrong Number");
27. }
28. **else**
29. Console.Write("Entered Number is not an Armstrong Number");
30. Console.ReadLine();
31. }
32. }
33. }

advertisements

Here is the output of the C# Program:

Enter the Number : 371

Entered Number is an Armstrong Number

C# Program to Check Whether the Entered Number is a Perfect Number or Not

This C# Program Checks Whether the Entered Number is a Perfect Number or Not. A perfect number is a positive integer that is equal to the sum of its proper divisors.

Here is source code of the C# Program to Check Whether the Entered Number is a Perfect Number or Not. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Check Whether the Entered Number is a Perfect Number or Not*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
9. **namespace** Program
10. {
11. **class** Program
12. {
13. **static** **void** Main(**string**[] args)
14. {
15. **int** number,sum=0,n;
16. Console.Write("enter the Number");
17. number = **int**.Parse(Console.ReadLine());
18. n = number;
19. **for** (**int** i = 1; i < number;i++)
20. {
21. **if** (number % i == 0)
22. {
23. sum=sum + i;
24. }
25. }
26. **if** (sum == n)
27. {
28. Console.WriteLine("**\n** Entered number is a perfect number");
29. Console.ReadLine();
30. }
31. **else**
32. {
33. Console.WriteLine("**\n** Entered number is not a perfect number");
34. Console.ReadLine();
35. }
36. }
37. }
38. }

advertisements

Here is the output of the C# Program:

Enter the Number : 6

Entered Number is a Perfect Number

C# Program to Perform all Basic Arithmetic Operations

This C# Program Performs all Basic Arithmetic Operations. Two numbers are obtained from the user and the operation which the user needs to perform.Based on the option entered by the user the arithmetic operation is performed.

Here is source code of the C# Program to Perform all Basic Arithmetic Operations. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Perform all Basic Arithmetic Operations*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Text;
7. **namespace** Program
8. {
9. **class** Program
10. {
11. **static** **void** Main(**string**[] args)
12. {
13. **int** Num1, Num2, result;
14. **char** option;
15. Console.Write("Enter the First Number : ");
16. Num1 = Convert.ToInt32(Console.ReadLine());
17. Console.Write("Enter the Second Number : ");
18. Num2 = Convert.ToInt32(Console.ReadLine());
19. Console.WriteLine("Main Menu");
20. Console.WriteLine("1. Addition");
21. Console.WriteLine("2. Subtraction");
22. Console.WriteLine("3. Multiplication");
23. Console.WriteLine("4. Division");
24. Console.Write("Enter the Operation you want to perform : ");
25. option = Convert.ToChar(Console.ReadLine());
26. **switch** (option)
27. {
28. **case** '1':
29. result = Num1 + Num2;
30. Console.WriteLine("The result of Addition is : {0}", result);
31. **break**;
32. **case** '2':
33. result = Num1 - Num2;
34. Console.WriteLine("The result of Subtraction is : {0}", result);
35. **break**;
36. **case** '3':
37. result = Num1 \* Num2;
38. Console.WriteLine("The result of Multiplication is : {0}", result);
39. **break**;
40. **case** '4':
41. result = Num1 / Num2;
42. Console.WriteLine("The result of Division is : {0}", result);
43. **break**;
44. **default**:
45. Console.WriteLine("Invalid Option");
46. **break**;
47. }
48. Console.ReadLine();
49. }
51. }
52. }

advertisements

Here is the output of the C# Program:

Enter the First Number : 100

Enter the Second Number : 2

Main Menu

1. Addition

2. Subtraction

3. Multiplication

4. Division

Enter the Operation you want to perform : 3

The Result of Multiplication is : 200

C# Program to Display the Factors of the Entered Number

This C# Program Displays the Factors of the Entered Number. The factors of a number are all those numbers that can divide evenly into the number with no remainder.

Here is source code of the C# program which checks a given integer is odd or even. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Display the Factors of the Entered Number*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
9. **namespace** Program
10. {
11. **class** Program
12. {
13. **static** **void** Main(**string**[] args)
14. {
15. **int** num, x;
16. Console.WriteLine("Enter the Number ");
17. num = **int**.Parse(Console.ReadLine());
18. Console.WriteLine("The Factors are : ");
19. **for** (x = 1; x <= num; x++)
20. {
21. **if** (num % x == 0)
22. {
23. Console.WriteLine(x);
24. }
25. }
26. Console.ReadLine();
28. }
29. }
30. }

advertisements

Here is the output of the C# Program:

Enter the Number : 27

The Factors are :

1

3

9

27

C# Program to Find and Display the H.C.F of a Given Number

This C# Program Finds and Display the H.C.F of a Given Number. In other words the H.C.F is the largest of all the common factors.

Here is source code of the C# Program to Find and Display the H.C.F of a Given Number. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Find and Display the H.C.F of a Given Number*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
9. **namespace** Program
10. {
11. **class** Program
12. {
13. **public** **static** **void** Main(**string**[] args)
14. {
15. **int** num1,num2,i;
16. **int** hcf =0;
17. Console.Write("**\n**Enter the First Number : ");
18. num1 = **int**.Parse(Console.ReadLine());
19. Console.Write("**\n**Enter the Second Number : ");
20. num2 = **int**.Parse(Console.ReadLine());
21. **for**(i=1;i<=num1||i<=num2;++i)
22. {
23. **if**(num1%i==0 && num2%i==0)
24. {
25. hcf=i;
26. }
27. }
28. Console.Write("**\n**Common Factor is : ");
29. Console.WriteLine(hcf);
30. Console.Read();
31. }
32. }
33. }

advertisements

Here is the output of the C# Program:

Enter the First Number : 12

Enter the Second Number : 16

Common Factor is : 4

C# Program to Check Whether the Entered Number is a Amicable Number or Not

This C# Program Checks Whether the Entered Number is a Amicable Number or Not. Amicable numbers are two numbers so related that the sum of the proper divisors of the one is equal to the other, unity being considered as a proper divisor but not the number itself.

Here is source code of the C# Program that Checks Whether the Entered Number is a Amicable Number or Not. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program Checks Whether the Entered Number is a Amicable Number or Not*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
9. **namespace** Program
10. {
11. **class** Program
12. {
13. **public** **static** **void** Main(**String**[] args)
14. {
15. **int** num1, num2, sum1 = 0, sum2 = 0, i;
16. Console.WriteLine("Enter First Number : ");
17. num1 = **int**.Parse(Console.ReadLine());
18. Console.WriteLine("Enter Second Number : ");
19. num2 = **int**.Parse(Console.ReadLine());
20. **for** (i = 1; i < num1; i++)
21. {
22. **if** (num1 % i == 0)
23. {
24. sum1 = sum1 + i;
25. }
26. }
27. **for** (i = 1; i < num2; i++)
28. {
29. **if** (num2 % i == 0)
30. {
31. sum2 = sum2 + i;
32. }
33. }
34. **if** (num1 == sum2 && num2 == sum1)
35. {
36. Console.WriteLine("They are a Pair of Amicable Numbers");
37. Console.ReadLine();
38. }
39. **else**
40. {
41. Console.WriteLine("They are not Amicable Numbers");
42. Console.ReadLine();
43. }
44. }
45. }
46. }

advertisements

Here is the output of the C# Program:

Enter First Number : 220

Enter Second Number :284

They are a Pair Of Amicable Numbers

#### 2. C# Examples on Complex Numbers and Trigonometric Functions

The Complex number is of the form a+ib where a is the real part and b is the imaginary part. The following programs cover various sections of Mathematics and Statistics. They are trigonometric functions such as sin(x) and cos(x), Mean, Standard Deviation for the given set of numbers and other applications related to Speed, Distance and Time.

C# Program to Add 2 Complex Numbers

This C# Program Adds two Complex Numbers.Group the real part of the complex number and the imaginary part of the complex number and then add.

Here is source code of the C# program that Adds two Complex Numbers.The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Add 2 Complex Numbers*
3. *\*/*
4. **using** System;
5. **public** **struct** Complex
6. {
7. **public** **int** real;
8. **public** **int** imaginary;
10. **public** Complex(**int** real, **int** imaginary)
11. {
12. **this**.real = real;
13. **this**.imaginary = imaginary;
14. }

17. **public** **static** Complex **operator** +(Complex c1, Complex c2)
18. {
19. **return** new Complex(c1.real + c2.real, c1.imaginary + c2.imaginary);
20. }

23. **public** **override** **string** ToString()
24. {
25. **return** (**String**.Format("{0} + {1}i", real, imaginary));
26. }
27. }
29. **class** TestComplex
30. {
31. **static** **void** Main()
32. {
33. Complex num1 = new Complex(2, 3);
34. Complex num2 = new Complex(3, 4);
35. Complex sum = num1 + num2;
36. Console.WriteLine("First Complex Number : {0}", num1);
37. Console.WriteLine("Second Complex Number : {0}", num2);
38. Console.WriteLine("The Sum of the Two Numbers : {0}", sum);
39. Console.ReadLine();
40. }
41. }

advertisements

Here is the output of the C# Program:

First Complex Number : 2+3i

Second Complex Number : 3+4i

The Sum of the Two Numbers : 5+7i

C# Program to Perform GCD

This C# Program Performs GCD. Greatest Common Divisor of two numbers is a largest positive numbers which can divide both numbers without any remainder

Here is source code of the C# Program that calculate GCD between two numbers. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Perform GCD*
3. *\*/*
4. **using** System;
5. **public** **class** Program
6. {
7. **static** **int** GCD(**int** num1, **int** num2)
8. {
9. **int** Remainder;
11. **while** (num2 != 0)
12. {
13. Remainder = num1 % num2;
14. num1 = num2;
15. num2 = Remainder;
16. }
18. **return** num1;
19. }
21. **static** **int** Main(**string**[] args)
22. {
23. **int** x, y;
24. Console.Write("Enter the First Number : ");
25. x = **int**.Parse(Console.ReadLine());
26. Console.Write("Enter the Second Number : ");
27. y = **int**.Parse(Console.ReadLine());
28. Console.Write("**\n**The Greatest Common Divisor of ");
29. Console.WriteLine("{0} and {1} is {2}", x, y, GCD(x, y));
30. Console.ReadLine();
31. **return** 0;
32. }
33. }

advertisements

Here is the output of the C# Program:

Enter the First Number : 12

Enter the Second Number : 24

The Greatest Common Divisor of 12 and 24 is : 12

C# Program to calculate the series sin(x)=x-x^3/3!+x^5/!-x^7/7!+……

This C# Program Finds the Value of sin(x) from the series sin(x)=x-x^3/3!+x^5/5-x^7/7!

Here is source code of the C# program to Find the Value of sin(x) from the Series. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to calculate the series sin(x)=x-x^3/3!+x^5/!-x^7/7!+......*
3. *\*/*
4. **using** System;
5. **class** sine
6. {
7. **int** deg, n;
8. **public** **void** readdata()
9. {
10. Console.WriteLine("Enter the Number of Terms:");
11. n = Convert.ToInt32(Console.ReadLine());
12. Console.WriteLine("Enter the Angle in Degrees:");
13. deg = Convert.ToInt32(Console.ReadLine());
14. }
15. **public** **void** sineseries()
16. {
17. **float** x, s = 0.0f, t;
18. x = (**float**)Math.PI \* deg / 180f;
19. s = x;
20. t = x;
21. **for** (**int** i = 1; i <= n; i++)
22. {
23. t = (-t \* x \* x) / ((2 \* i) \* (2 \* i + 1));
24. s = s + t;
25. }
26. Console.WriteLine("Sin({0})={1}", deg, s);
27. }
28. **public** **static** **void** Main()
29. {
30. sine s = new sine();
31. s.readdata();
32. s.sineseries();
33. }
34. }

advertisements

Here is the output of the C# Program:

Enter the Number of Terms:

20

Enter the Angle in Degrees:

90

Sin(90)=0.99999994

C# Program to Illustrate Trignometry Angles in Degrees

This C# Program Illustrates Trignometry Angles in Degrees. Here the trigonometric values are calculated in terms of radians and are displayed.

Here is source code of the C# Program to Illustrate Trignometry Angles in Degrees. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Illustrate Trignometry Angles in Degrees*
3. *\*/*
4. **using** System;
5. **namespace** trig
6. {
7. **class** Program
8. {
9. **static** **void** Main(**string**[] args)
10. {
11. Console.WriteLine("Trignometric values in Degree");
12. Console.WriteLine("sin (60) = {0}", Math.Sin(60 \* Math.PI / 180));
13. Console.WriteLine("cos (60) = {0}", Math.Cos(60 \* Math.PI / 180));
14. Console.WriteLine("tan (60) = {0}", Math.Tan(60 \* Math.PI / 180));
15. Console.WriteLine("arcsin (1/2) = {0}", Math.Asin(0.5) \* 180 / Math.PI);
16. Console.WriteLine("arccos (1/2) = {0}", Math.Acos(0.5) \* 180 / Math.PI);
17. Console.WriteLine("arctan (1/2) = {0}", Math.Atan(0.5) \* 180 / Math.PI);
18. Console.Read();
19. }
20. }
21. }

advertisements

Here is the output of the C# Program:

Trignometric Values in Radians :

sin (pi/3) = 0.866025403784439

cos (pi/3) = 0.5

tan (pi/3) = 1.73205080756888

arcsin (1/2) = 30

arccos (1/2) = 60

arctan (1/2) = 26.565051177078

C# Program to Illustrate Trignometry Angles in Radians

This C# Program Illustrates Trignometry Angles in Radians. Here the trigonometric values are calculated in terms of radians and are displayed.

Here is source code of the C# Program to Illustrate Trignometry Angles in Radians. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Illustrate Trignometry Angles in Radians*
3. *\*/*
4. **using** System;
5. **namespace** trig
6. {
7. **class** Program
8. {
9. **static** **void** Main(**string**[] args)
10. {
11. Console.WriteLine("Trignometric Values in Radians : ");
12. Console.WriteLine("sin (pi/3) = {0}", Math.Sin(Math.PI / 3));
13. Console.WriteLine("cos (pi/3) = {0}", Math.Cos(Math.PI / 3));
14. Console.WriteLine("tan (pi/3) = {0}", Math.Tan(Math.PI / 3));
15. Console.WriteLine("arcsin (1/2) = {0}", Math.Asin(0.5));
16. Console.WriteLine("arccos (1/2) = {0}", Math.Acos(0.5));
17. Console.WriteLine("arctan (1/2) = {0}", Math.Atan(0.5));
18. Console.ReadLine();
19. }
20. }
21. }

advertisements

Here is the output of the C# Program:

Trignometric Values in Radians :

sin (pi/3) = 0.866025403784439

cos (pi/3) = 0.5

tan (pi/3) = 1.73205080756888

arcsin (1/2) = 0.523598775598299

arccos (1/2) = 1.0471975511966

arctan (1/2) = 0.463647609000806

C# Program to Find and display the Multiplication Table

This C# Program Finds and display the Multiplication Table. Here the limit is obtained from the user and the multiplication table is diaplayed.

Here is source code of the C# Program to Find and display the Multiplication Table. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Find and display the Multiplication Table*
3. *\*/*
4. **using** System;
5. **class** Multipication
6. {
7. **static** **void** Main()
8. {
9. **int** no;
11. Console.Write("Enter a no : ");
12. no = Convert.ToInt32(Console.ReadLine());
13. **while** (no <= 0)
14. {
15. Console.WriteLine("You entered an invalid no");
17. Console.Write("Enter a no great than 0: ");
18. no = Convert.ToInt32(Console.ReadLine());
19. }
20. Console.WriteLine("Multiplication Table :");
21. **for** (**int** i = 1; i <= no; i++)
22. {
23. Console.WriteLine("**\n**");
25. **for** (**int** j = 1; j <= no; j++)
26. {
27. Console.Write("{0,6}", i \* j);
28. }
30. }
31. Console.Read();
32. }
33. }

advertisements

Here is the output of the C# Program:

Enter a No : 5

Multiplication Table :

1 2 3 4 5

2 4 6 8 10

3 6 9 12 15

4 8 12 16 20

5 10 15 20 25

C# Program to Find the Standard Deviation of a Set of Given Numbers

This C# Program Finds the Standard Deviation of a Set of Given Numbers. Here the mean,variance and standard deviation is calculated and displayed.

Here is source code of the C# Program to Find the Standard Deviation of a Set of Given Numbers. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Find the Standard Deviation of a Set of Given Numbers*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **namespace** SampleApp
7. {
8. **internal** **class** Program
9. {
10. **private** **static** **void** Main()
11. {
12. List<**double**> number = new List<**double**> { 1, 2, 3, 4, 5, 6 };
14. **double** mean = number.Mean();
15. **double** variance = number.Variance();
16. **double** sd = number.StandardDeviation();
18. Console.WriteLine("Mean: {0} , Variance: {1} , SD: {2} ", mean, variance, sd);
19. Console.ReadKey();
20. }
21. }
22. **public** **static** **class** list
23. {
24. **public** **static** **double** Mean(**this** List<**double**> values)
25. {
26. **return** values.Count == 0 ? 0 : values.Mean(0, values.Count);
27. }
29. **public** **static** **double** Mean(**this** List<**double**> values, **int** start, **int** end)
30. {
31. **double** s = 0;
33. **for** (**int** i = start; i < end; i++)
34. {
35. s += values[i];
36. }
38. **return** s / (end - start);
39. }
41. **public** **static** **double** Variance(**this** List<**double**> values)
42. {
43. **return** values.Variance(values.Mean(), 0, values.Count);
44. }
46. **public** **static** **double** Variance(**this** List<**double**> values, **double** mean)
47. {
48. **return** values.Variance(mean, 0, values.Count);
49. }
51. **public** **static** **double** Variance(**this** List<**double**> values, **double** mean, **int** start, **int** end)
52. {
53. **double** variance = 0;
55. **for** (**int** i = start; i < end; i++)
56. {
57. variance += Math.Pow((values[i] - mean), 2);
58. }
60. **int** n = end - start;
61. **if** (start > 0) n -= 1;
63. **return** variance / (n);
64. }
65. **public** **static** **double** StandardDeviation(**this** List<**double**> values)
66. {
67. **return** values.Count == 0 ? 0 : values.StandardDeviation(0, values.Count);
68. }
69. **public** **static** **double** StandardDeviation(**this** List<**double**> values, **int** start, **int** end)
70. {
71. **double** mean = values.Mean(start, end);
72. **double** variance = values.Variance(mean, start, end);
73. **return** Math.Sqrt(variance);
74. }
75. }
76. }

advertisements

Here is the output of the C# Program:

Mean : 3.5 Variance : 2.916666666667 S.D = 1.7078251256993

C# Program to Find the Mean of given Set of Numbers

This C# Program Finds the Mean of given Set of Numbers. Here the sum and average of five numbers are found and displayed.

Here is source code of the C# Program to Find the Mean of given Set of Numbers. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Find the Mean of given Set of Numbers*
3. *\*/*
4. **using** System;
5. **class** avg
6. {
7. **public** **static** **void** Main()
8. {
9. **int** n1, n2, n3, n4, n5, avg, sum;
10. Console.WriteLine("Enter 5 Numbers:");
11. n1 = Convert.ToInt32(Console.ReadLine());
12. n2 = Convert.ToInt32(Console.ReadLine());
13. n3 = Convert.ToInt32(Console.ReadLine());
14. n4 = Convert.ToInt32(Console.ReadLine());
15. n5 = Convert.ToInt32(Console.ReadLine());
16. sum = (n1 + n2 + n3 + n4 + n5);
17. avg = (sum / 5);
18. Console.WriteLine("Sum :" + sum);
19. Console.WriteLine("Average :" + avg);
20. Console.ReadLine();
21. }
22. }

advertisements

Here is the output of the C# Program:

Enter 5 Numbers :

10

10

10

10

10

Sum : 50

Average : 10

C# Program to Calculate Simple Interest

This C# Program Calculates Simple Interest. Here Simple interest is determined by multiplying the interest rate by the principal by the number of periods.

Here is source code of the C# Program to Calculate Simple Interest. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Calculate Simple Interest*
3. *\*/*
4. **using** System;
5. **namespace** Interest
6. {
7. **class** Program
8. {
9. **static** **void** Main(**string**[] args)
10. {
11. **int** year;
12. **double** princamt,rate, interest, total\_amt;
13. Console.Write("Enter The Loan Amount : ");
14. princamt = Convert.ToDouble(Console.ReadLine());
15. Console.Write("Enter The Number of Years : ");
16. year = Convert.ToInt16(Console.ReadLine());
17. Console.Write("Enter the Rate Of Interest : ");
18. rate = Convert.ToDouble(Console.ReadLine());
19. interest = princamt \* year \* rate / 100;
20. total\_amt = princamt + interest;
21. Console.WriteLine("Total Amount : {0}", total\_amt);
22. Console.ReadLine();
23. }
24. }
25. }

advertisements

Here is the output of the C# Program:

Enter the Loan Amount : 1000

Enter the Number of Years : 3

Enter the Rate of Interest : 2

Total Amount : 1060

C# Program to Calculate Compound Interest

This C# Program Calculates Compound Interest. Compounding of interest allows a principal amount to grow at a faster rate than simple interest, which is calculated as a percentage of only the principal amount.

Here is source code of the C# Program to Calculate Compound Interest. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Calculate Compound Interest*
3. *\*/*
4. **using** System;
5. **namespace** compund
6. {
7. **class** compound
8. {
9. **static** **void** Main(**string**[] args)
10. {
11. **double** Total = 0, interestRate, years, annualCompound, Amount;
12. Console.Write("Enter the Initial Amount : ");
13. Amount = Convert.ToDouble(Console.ReadLine());
14. Console.Write("Enter the Rate of Interest : ");
15. interestRate = Convert.ToDouble(Console.ReadLine()) / 100;
16. Console.Write("Enter the Number of Years : ");
17. years = Convert.ToDouble(Console.ReadLine());
18. Console.Write("Number of Times the Interest will be Compounded : ");
19. annualCompound = Convert.ToDouble(Console.ReadLine());
20. **for** (**int** t = 1; t < years + 1; t++)
21. {
22. Total = Amount \* Math.Pow((1 + interestRate / annualCompound), (annualCompound \* t));
23. Console.Write("Your Total for Year {0} "
24. + "is {1:F0}. **\n**", t, Total);
26. }
28. Console.ReadLine();
29. }
30. }
31. }

advertisements

Here is the output of the C# Program:

Enter the Initial Amount : 1000

Enter the Rate of Interest : 2

Enter the Number of Years : 2

Number of Times the Interest will be Compounded : 2

Your Total for Year 1 is : 1020

Your Total for Year 2 is : 1041

C# Program to Find the Value of Cos(x)

This C# Program Finds the Value of Cos(x).Here the cos value is found without using the library math function.

Here is source code of the C# Program to Find the Value of Cos(x) . The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Find the Value of Cos(x)*
3. *\*/*
4. **using** System;
5. **namespace** ConsoleApplication1
6. {
7. **class** Program
8. {
9. **static** **void** Main(**string**[] args)
10. {
11. **for** (**double** d = 0; d < 6.0; d += 0.5)
12. {
13. Console.WriteLine("The cosine of {0} = {1}", d, Math.Cos(d));
14. Console.WriteLine("Calculated cosine of {0} = {1}", d, cos(d));
15. Console.WriteLine();
16. }
17. Console.ReadKey();
18. }
20. **static** **double** cos(**double** x)
21. {
22. **double** p = x \* x;
23. **double** q = p \* p;
24. **return** 1.0 - p / 2 + q / 24 - p \* q / 720 + q \* q / 40320 - p \* q \* q / 3628800;
25. }
26. }
27. }

advertisements

Here is the output of the C# Program:

The Cosine of 0 = 1

Calculated Cosine of 0 = 1

The Cosine of 0.5 = 0.877582561890373

Calculated Cosine of 0.5 = 0.877582561889864

The Cosine of 1.5 = 0.54030230586814

Calculated Cosine of 1.5 = 0.540302303791887

The Cosine of 2.0 = 0.0707372016677029

Calculated Cosine of 2.0 = 0.0707369341169085

C# Program to Reverse a Number & Check if it is a Palindrome

This C# Program Reverses a Number & Check if it is a Palindrome. Here First it reverses a number. Then it checks if given number and reversed numbers are equal. If they are equal, then its a palindrome.

Here is source code of the C# Program to Reverse a Number & Check if it is a Palindrome. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Reverse a Number & Check if it is a Palindrome*
3. *\*/*
4. **using** System;
5. **class** program
6. {
7. **public** **static** **void** Main()
8. {
9. **int** num, temp, remainder, reverse = 0;
10. Console.WriteLine("Enter an integer **\n**");
11. num = **int**.Parse(Console.ReadLine());
12. temp = num;
13. **while** (num > 0)
14. {
15. remainder = num % 10;
16. reverse = reverse \* 10 + remainder;
17. num /= 10;
18. }
19. Console.WriteLine("Given number is = {0}", temp);
20. Console.WriteLine("Its reverse is = {0}", reverse);
21. **if** (temp == reverse)
22. Console.WriteLine("Number is a palindrome **\n**");
23. **else**
24. Console.WriteLine("Number is not a palindrome **\n**");
25. Console.ReadLine();
26. }
27. }

advertisements

Here is the output of the C# Program:

Enter an integer

343

Given number is = 343

Its reverse is = 343

Number is a palindrome

C# Program to Calculate the Distance Travelled by Reading Speed and Time

This C# Program Calculates the Distance Travelled by Reading Speed and Time. Here distance is calculated by multiplying speed and time.

Here is source code of the C# Program to Calculate the Distance Travelled by Reading Speed and Time. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Calculate the Distance Travelled by Reading Speed and Time*
3. *\*/*
4. **using** System;
5. **class** program
6. {
7. **public** **static** **void** Main()
8. {
9. **int** speed, distance, time;
10. Console.WriteLine("Enter the Speed(km/hr) : ");
11. speed = Convert.ToInt32(Console.ReadLine());
12. Console.WriteLine("Enter the Time(hrs) : ");
13. time = Convert.ToInt32(Console.ReadLine());
14. distance = speed \* time;
15. Console.WriteLine("Distance Travelled (kms) : " + distance);
16. Console.ReadLine();
17. }
18. }

advertisements

Here is the output of the C# Program:

Enter the Speed(km/hr) : 5

Enter the Time(hrs) : 4

Distance Travelled (kms) : 20

#### 3. C# Examples on Divisibility Tests and Bitwise Operations

The following section contains programs on different types of mathematical numbers. They are Prime numbers, Binary numbers, Natural numbers, Divisibility tests for different numbers. Apart from this it contains programs to implement special types of triangles like Floyd triangle and Pascal triangle. It performs Addition, Subtraction, Multiplication and Summation operations on the above mentioned special numbers. The remaining programs include demonstration of Bitwise operations and calculating the powers of various numbers.

C# Program to Check Whether the Given Number is a Prime number if so then Display its Largest Facor

This C# Program Checks Whether the Given Number is a Prime number if so then Display its Largest Facor. Here first the number that is obtained is checked whether the number is prime or not and then the largest factor of it is displayed.

Here is source code of the C# Program to Check Whether the Given Number is a Prime number if so then Display its Largest Facor. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Check Whether the Given Number is a Prime number if so then*
3. *\* Display its Largest Factor*
4. *\*/*
5. **using** System;
6. **namespace** example
7. {
8. **class** prime
9. {
10. **public** **static** **void** Main()
11. {
12. Console.Write("Enter a Number : ");
13. **int** num;
14. num = Convert.ToInt32(Console.ReadLine());
15. **int** k;
16. k = 0;
17. **for** (**int** i = 1; i <= num; i++)
18. {
19. **if** (num % i == 0)
20. {
21. k++;
22. }
23. }
24. **if** (k == 2)
25. {
26. Console.WriteLine("Entered Number is a Prime Number and the Largest Factor is {0}",num);
27. }
28. **else**
29. {
30. Console.WriteLine("Not a Prime Number");
31. }
32. Console.ReadLine();
33. }
34. }
35. }

advertisements

Here is the output of the C# Program:

Enter a Number : 23

Entered Number is a Prime Number and the Largest Factor is 23

C# Program to Illustrate Pascal Triangle

This C# Program Illustrates Pascal Triangle. Here This program uses the for loops to print the Pascal’s triangle.

Here is source code of the C# Program to Illustrate Pascal Triangle. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Illustrate Pascal Triangle*
3. *\*/*
4. **using** System;
5. **class** Pascal
6. {
7. **public** **static** **void** Main()
8. {
9. **int**[,] arr = new **int**[8, 8];
10. Console.WriteLine("Pascal Triangle : ");
11. **for** (**int** i = 0; i < 5; i++)
12. {
13. **for** (**int** k = 5; k > i; k--)
14. {
15. Console.Write(" ");
16. }
18. **for** (**int** j = 0; j < i; j++)
19. {
20. **if** (j == 0 || i == j)
21. {
22. arr[i, j] = 1;
23. }
24. **else**
25. {
26. arr[i, j] = arr[i - 1, j] + arr[i - 1, j - 1];
27. }
28. Console.Write(arr[i, j] + " ");
29. }
30. Console.ReadLine();
31. }
32. }
33. }

advertisements

Here is the output of the C# Program:

Pascal Triangle :

1

1 1

1 2 1

1 3 3 1

C# Program to Find the Sum of first 50 Natural Numbers using For Loop

This C# Program Finds the Sum of first 50 Natural Numbers using For Loop.

Here is source code of the C# Program to Find the Sum of first 50 Natural Numbers using For Loop. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Find the Sum of first 50 Natural Numbers*
3. *\* using For Loop*
4. *\*/*
5. **using** System;
6. **class** program
7. {
8. **public** **static** **void** Main()
9. {
10. **int** num, sum = 0;
11. **for** (num = 1; num <= 50; num++)
12. {
13. sum = sum + num;
14. }
15. Console.WriteLine("Sum = {0}", sum);
16. Console.ReadLine();
17. }
18. }

advertisements

Here is the output of the C# Program:

Sum = 1275

C# Program to Multiply given Number by 4 using Bitwise Operators

This C# Program Multiplies given Number by 4 using Bitwise Operators. Here The bitwise operators are or, and, xor, not, left shift, right shift. Program uses left shift operator for this.

Here is source code of the C# Program to Multiply given Number by 4 using Bitwise Operators. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Multiply given Number by 4 using Bitwise Operators*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
9. **namespace** ConsoleApplication
10. {
11. **class** Program
12. {
13. **static** **void** Main(**string**[] args)
14. {
15. **int** number, tempnum;
16. Console.WriteLine("Enter an integer :");
17. number = **int**.Parse(Console.ReadLine());
18. tempnum = number;
19. number = number << 2;
20. Console.WriteLine("{0},{1}", tempnum, number);
21. Console.ReadLine();
22. }
23. }
24. }

advertisements

Here is the output of the C# Program:

Enter an integer :

120

120,480

C# Program to Find the Sum of two Binary Numbers

This C# Program Finds the Sum of two Binary Numbers. Here Binary number is a number that can be represented using only two numeric symbols – 0 and 1. A number in base 2.

Here is source code of the C# Program to Find the Sum of two Binary Numbers. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Find the Sum of two Binary Numbers \*/*
3. **using** System;
4. **using** System.Collections.Generic;
5. **using** System.Linq;
6. **using** System.Text;
7. **namespace** ConsoleApplication
8. {
9. **class** Program
10. {
11. **static** **void** Main(**string**[] args)
12. {
13. **int** b1, b2;
14. **int** i = 0, rem = 0;
15. **int**[] sum = new **int**[20];
16. Console.WriteLine("Enter the first binary number: ");
17. b1 = **int**.Parse(Console.ReadLine());
18. Console.WriteLine("Enter the second binary number: ");
19. b2 = **int**.Parse(Console.ReadLine());
20. **while** (b1 != 0 || b2 != 0)
21. {
22. sum[i++] = (b1 % 10 + b2 % 10 + rem) % 2;
23. rem = (b1 % 10 + b2 % 10 + rem) / 2;
24. b1 = b1 / 10;
25. b2 = b2 / 10;
26. }
27. **if** (rem != 0)
28. sum[i++] = rem;
29. --i;
30. Console.WriteLine("Sum of two binary numbers: ");
31. **while** (i >= 0)
32. Console.Write("{0}", sum[i--]);
33. Console.ReadLine();
34. }
35. }
36. }

advertisements

Here is the output of the C# Program:

Enter the first binary number:

100

Enter the second binary number:

110

Sum of two binary numbers:

1010

C# Program to Display Floyd’s Triangle with an Numeric Mode

This C# Program Displays Floyd’s Triangle with an Numeric Mode. Here the numbers are displayed in the form of a triangle.

Here is source code of the C# Program to Display Floyd’s Triangle with an Numeric Mode. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Display Floyd's Triangle with an Numeric Mode*
3. *\*/*
4. **using** System;
5. **class** Program
6. {
7. **static** **void** Main(**string**[] args)
8. {
10. **int** i, j, k = 1;
11. **for** (i = 1; i <= 10; i++)
12. {
13. **for** (j = 1; j < i + 1; j++)
14. {
15. Console.Write(k++ + " ");
16. }
18. Console.Write("**\n**");
19. }
20. Console.ReadLine();
21. }
22. }

advertisements

Here is the output of the C# Program:

1

2 3

4 5 6

7 8 9 10

11 12 13 14 15

16 17 18 19 20 21

22 23 24 25 26 27 28

29 30 31 32 33 34 35 36

37 38 39 40 41 42 43 44 45

46 47 48 49 50 51 52 53 54 55

C# Program to Find Multiplication of two Binary Numbers

This C# Program Finds Multiplication of two Binary Numbers. Here Binary number is a number that can be represented using only two numeric symbols – 0 and 1. A number in base 2. This program multiplies the 2 binary numbers.

Here is source code of the C# Program to Find Multiplication of two Binary Numbers. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Find Multiplication of two Binary Numbers*
3. *\*/*
4. **using** System;
5. **class** program
6. {
7. **public** **static** **void** Main()
8. {
10. **int** binary1, binary2, multiply = 0;
11. **int** digit, factor = 1;
12. prog pg = new prog();
13. Console.WriteLine("Enter the first binary number: ");
14. binary1 = **int**.Parse(Console.ReadLine());
15. Console.WriteLine("Enter the second binary number: ");
16. binary2 = **int**.Parse(Console.ReadLine());
17. **while** (binary2 != 0)
18. {
19. digit = binary2 % 10;
20. **if** (digit == 1)
21. {
22. binary1 = binary1 \* factor;
23. multiply = pg.binaryproduct(binary1, multiply);
24. }
25. **else** {
26. binary1 = binary1 \* factor;
27. binary2 = binary2 / 10;
28. factor = 10;
29. }
30. Console.WriteLine("Product of two binary numbers: {0}", multiply);
31. Console.ReadLine();
32. }
33. }
34. **class** prog
35. {
36. **public** **int** binaryproduct(**int** binary1, **int** binary2)
37. {
38. **int** i = 0, remainder = 0;
39. **int**[] sum = new **int**[20];
40. **int** binaryprod = 0;
41. **while** (binary1 != 0 || binary2 != 0)
42. {
43. sum[i++] =(binary1 % 10 + binary2 % 10 + remainder) % 2;
44. remainder =(binary1 % 10 + binary2 % 10 + remainder) / 2;
45. binary1 = binary1 / 10;
46. binary2 = binary2 / 10;
47. }
48. **if** (remainder != 0)
49. sum[i++] = remainder;
50. --i;
51. **while** (i >= 0)
52. binaryprod = binaryprod \* 10 + sum[i--];
53. **return** binaryprod;
54. }
55. }

advertisements

Here is the output of the C# Program:

Enter the first binary number : 1010

Enter the second binary number : 1011

Product of two binary numbers : 1101110

C# Program to Perform Multiplication of Exponents of Same Base

This C# Program Performs Multiplication of Exponents of Same Base. Here the power value for all same base is found.

Here is source code of the C# Program to Perform Multiplication of Exponents of Same Base. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Perform Multiplication of Exponents of Same Base*
3. *\*/*
4. **using** System;
5. **class** Program
6. {
7. **static** **void** Main()
8. {
9. Console.WriteLine("Enter the Base : ");
10. **double** num = **double**.Parse(Console.ReadLine());
11. Console.WriteLine("Enter the First Exponent :");
12. **double** exp1 = **double**.Parse(Console.ReadLine());
13. Console.WriteLine("Enter the Second Exponent :");
14. **double** exp2 = **double**.Parse(Console.ReadLine());
15. **double** mul;
16. mul = exp1 + exp2;
17. Console.WriteLine("Result is : {0}^{1} : {2}", num, mul, Math.Pow(num, mul));
18. Console.ReadLine();
19. }
20. }

advertisements

Here is the output of the C# Program:

Enter the Base :

2

Enter the First Exponent :

3

Enter the Second Exponent :

2

Result is : 2^5 : 32

C# Program to Check the Edge Values in Power Function

This C# Program Checks the Edge Values in Power Function. Here the power value for all the end values of the given number is found.

Here is source code of the C# Program to Check the Edge Values in Power Function. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Check the Edge Values in Power Function*
3. *\*/*
4. **using** System;
5. **class** Program
6. {
7. **static** **void** Main()
8. {
9. **double** value1 = Math.Pow(**double**.MinValue, **double**.MaxValue);
10. **double** value2 = Math.Pow(**double**.MinValue, 0);
11. **double** value3 = Math.Pow(**double**.NaN, 2);
12. **double** value4 = Math.Pow(**double**.PositiveInfinity, 2);
13. **double** value5 = Math.Pow(**double**.NegativeInfinity, 2);
14. Console.WriteLine("Result : {0}", value1);
15. Console.WriteLine("Result : {0}", value2);
16. Console.WriteLine("Result : {0}", value3);
17. Console.WriteLine("Result : {0}", value4);
18. Console.WriteLine("Result : {0}", value5);
19. Console.ReadLine();
20. }
21. }

advertisements

Here is the output of the C# Program:

Result : Infinity

Result : 1

Result : NaN

Result : Infinity

Result : Infinity

C# Program to Calculate Fractional Power

This C# Program Calculates Fractional Power. Here the power value of the given number is found.

Here is source code of the C# Program to Calculate Fractional Powers. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Calculate Fractional Powers*
3. *\*/*
4. **using** System;
5. **class** Program
6. {
7. **static** **void** Main()
8. {
9. **double** value1 = Math.Pow(2, 2.1);
10. **double** value2 = Math.Pow(Math.E, 2);
11. **double** value3 = Math.Pow(Math.PI, 1);
12. Console.WriteLine("Result : {0}", value1);
13. Console.WriteLine("Result : {0}", value2);
14. Console.WriteLine("Result : {0}", value3);
15. Console.ReadLine();
16. }
17. }

advertisements

Here is the output of the C# Program:

Result : 4.28709385014517

Result : 7.38905609893065

Result : 3.14159265358979

C# Program to Calculate the Power Exponent Value

This C# Program Calculates the power exponent value. Here the power value of the given number is found.

Here is source code of the C# Program to Calculate the power exponent value. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Calculate the power exponent value*
3. *\*/*
4. **using** System;
5. **class** Program
6. {
7. **static** **void** Main()
8. {
10. **double** m, n;
11. Console.WriteLine("Enter the Number : ");
12. m = **double**.Parse(Console.ReadLine());
13. Console.WriteLine("Enter the Exponent : ");
14. n = **double**.Parse(Console.ReadLine());
15. **double** value1 = Math.Pow(m, n);
16. Console.WriteLine("Result : {0}", value1);
17. Console.ReadLine();
18. }
19. }

advertisements

Here is the output of the C# Program:

Enter the Number : 2

Enter the Exponent :3

Result : 8

C# Program to Calculate Period Duration

This C# Program Calculates Period Duration. Here the period duration is found.

Here is source code of the C# Program to Calculate Period Duration. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Calculate Period Duration*
3. *\*/*
4. **using** System;
5. **class** CompareDates
6. {
8. **public** **static** **void** Main()
9. {
10. DateTime today = DateTime.Now;
11. DateTime yesterday = today - new TimeSpan(1, 0, 0, 0);
12. DateTime tomorrow = today + new TimeSpan(1, 0, 0, 0);
13. Console.WriteLine("Yesterday was {0}", yesterday);
14. Console.WriteLine("Today is {0}", today);
15. Console.WriteLine("Tomorrow will be {0}", tomorrow);
16. Console.WriteLine("**\n**Is yesterday less than today? {0}.",
17. yesterday < today);
18. Console.WriteLine("Is today the same as tomorrow ? {0}.",
19. today == tomorrow);
21. TimeSpan totalTimespan = new TimeSpan(3, 5, 24, 17) +
22. new TimeSpan(1, 18, 35, 43);
23. Console.WriteLine(
24. "**\n**The length of the period is {0} days {1} hours" +
25. " {2} minutes {3} seconds.",
26. totalTimespan.Days, totalTimespan.Hours,
27. totalTimespan.Minutes, totalTimespan.Seconds);
28. Console.ReadLine();
29. }
31. }

advertisements

Here is the output of the C# Program:

Yesterday was 09-06-2014 15:52:34

Today is 10-06-2014 15:52:34

Tomorrow will be 11-06-2014 15:52:34

Is yesterday less than today? True.

Is today the same as tomorrow ? False.

The length of the period is 5 days 0 hours 0 minutes 0 seconds.

C# Program to Calculate Power of Three

This C# Program Calculates Power of Three. Here the power value of the given number is found.

Here is source code of the C# Program to Calculate Power of Three. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Calculate Power of Three*
3. *\*/*

6. **using** System;
8. **class** Program
9. {
10. **static** **void** Main(**string**[] args)
11. {
12. new GeneratePowers().RaiseToPower
13. (5, *// 4 values per table*
14. 3);
15. Console.ReadLine();*// power to raise each value*
16. }
17. }
19. **public** **class** GeneratePowers
20. {
21. **public** **void** RaiseToPower(**int** maxIterations, **int** power)
22. {
23. Console.WriteLine("{0,8}{1,16}",
24. "Number", "Power of " + power);
25. **for** (**int** i = 1; i <= maxIterations; ++i)
26. {
27. Console.Write("{0,5}{1,15}**\n**", i,
28. Math.Pow(i, power));
29. }
30. }
31. }

advertisements

Here is the output of the C# Program:

Number Power of 3

1 1

2 8

3 27

4 64

5 125

C# Program to Find whether the Number is Divisible by 2

This C# Program Finds whether the Number is Divisible by 2. Any whole number that ends in 0, 2, 4, 6, or 8 will be divisible by 2.Here the divisibility test is done by performing the mod function with 2.

Here is source code of the C# Program to Find whether the Number is Divisible by 2. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Find whether the Number is Divisible by 2*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
8. **using** System.Threading.Tasks;
10. **namespace** ConsoleApplication16
11. {
12. **class** Program
13. {
14. **static** **void** Main(**string**[] args)
15. {
16. **int** n;
17. Console.WriteLine("Enter the Number :");
18. n = **int**.Parse(Console.ReadLine());
19. **if** (n % 2 == 0)
20. {
21. Console.WriteLine("Entered Number is Divisible by 2 ");
22. }
23. **else**
24. {
25. Console.WriteLine("Entered Number is Not Divisible by 2");
26. }
27. Console.ReadLine();
28. }
29. }
30. }

advertisements

Here is the output of the C# Program:

Enter the Number :

45

Entered Number is Not Divisible by 2

C# Program to Illustrate Bitwise Operations

This C# Program Illustrates Bitwise Operations. Here there are many bitwise operators like AND,OR,XOR which performs corresponding operations.

Here is source code of the C# Program to Illustrate Bitwise Operations. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Illustrate Bitwise Operations*
3. *\*/*
4. **using** System;
5. **class** bitwise
6. {
7. **byte** b1, b2;
8. **int** x;
9. **long** y;
10. bitwise()
11. {
12. b1 = 10;
13. b2 = 5;
14. x = 32;
15. y = 20;
16. }
17. **public** **static** **void** Main()
18. {
19. bitwise bit = new bitwise();
20. **byte** p = (**byte**)(bit.b1 & bit.b2);
21. **byte** q = (**byte**)(bit.b1 | bit.b2);
22. **byte** r = (**byte**)(bit.b1 ^ bit.b2);
23. **int** z = (**int**)(bit.x & bit.y);
24. Console.WriteLine("b1={0},b2={1},x={2},y={3}", bit.b1, bit.b2, bit.x, bit.y);
25. Console.WriteLine("b1 & b2={0} : ", p);
26. Console.WriteLine("b1 | b2={0} : ", q);
27. Console.WriteLine("b1 ^ b2={0} : ", r);
28. Console.WriteLine("x & y = {0} : ", z);
29. Console.ReadLine();
30. }
31. }

advertisements

Here is the output of the C# Program:

b1=10,b2=5,x=32,y=20

b1 & b2 : 0

b1 | b2 : 15

b1 ^ b2 : 15

x & y :0

C# Program to Find Power of 2 using Bitwise Operator

This C# Program Finds Power of 2 using Bitwise Operator. Here Operations on bits at individual levels can be carried out using Bitwise operations and the whole representation of a number is considered while applying a bitwise operator.

Here is source code of the C# Program to Find Power of 2 using Bitwise Operator. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Find Power of 2 using Bitwise Operator*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Globalization;
7. **using** System.Linq;
8. **using** System.Text;
10. **namespace** example
11. {
12. **internal** **class** Program
13. {
14. **private** **static** **void** Main(**string**[] args)
15. {
16. **int** num;
17. Console.Write("Enter a number:");
18. num = Convert.ToInt32(Console.ReadLine());
19. **bool** result = ((num & -num) == num);
20. Console.WriteLine(result);
21. Console.ReadLine();
22. }
23. }
24. }

advertisements

Here is the output of the C# Program:

Enter a number:16

True

#### 4. C# Examples on Quadratic Equations

The equation which is of the form ax2+bx+c is called a Quadratic Equation. The distance of the number from 0 on a number line is called Absolute value of a number. The following programs evaluates the roots of a given quadratic equation, prints the absolute value of a given number, demonstrates the usage of static constructor and evaluates the LCM of the given numbers. The constructor that is called automatically even before the first instance of the class is created is called a static constructor.

C# Program to Find Roots of a Quadratic Equation

This C# Program Finds Roots of a Quadratic Equation. Here A quadratic equation is a second-order polynomial equation expressed in a single variable, x, with a ≠ 0: ax2+bx+c=0 and has two roots which is found and displayed.

Here is source code of the C# Program to Find Roots of a Quadratic Equation. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Find Roots of a Quadratic Equation*
3. *\*/*
4. **using** System;
6. **namespace** example
7. {
8. **class** Quadraticroots
9. {
10. **double** a, b, c;
12. **public** **void** read()
13. {
14. Console.WriteLine(" **\n** To find the roots of a quadratic equation of the form a\*x\*x + b\*x + c = 0");
15. Console.Write("**\n** Enter value for a : ");
16. a = **double**.Parse(Console.ReadLine());
17. Console.Write("**\n** Enter value for b : ");
18. b = **double**.Parse(Console.ReadLine());
19. Console.Write("**\n** Enter value for c : ");
20. c = **double**.Parse(Console.ReadLine());
21. }
22. **public** **void** compute()
23. {
24. **int** m;
25. **double** r1, r2, d1;
26. d1 = b \* b - 4 \* a \* c;
27. **if** (a == 0)
28. m = 1;
29. **else** **if** (d1 > 0)
30. m = 2;
31. **else** **if** (d1 == 0)
32. m = 3;
33. **else**
34. m = 4;
35. **switch** (m)
36. {
37. **case** 1: Console.WriteLine("**\n** Not a Quadratic equation, Linear equation");
38. Console.ReadLine();
39. **break**;
40. **case** 2: Console.WriteLine("**\n** Roots are Real and Distinct");
41. r1 = (-b + Math.Sqrt(d1)) / (2 \* a);
42. r2 = (-b - Math.Sqrt(d1)) / (2 \* a);
43. Console.WriteLine("**\n** First root is {0:#.##}", r1);
44. Console.WriteLine("**\n** Second root is {0:#.##}", r2);
45. Console.ReadLine();
46. **break**;
47. **case** 3: Console.WriteLine("**\n** Roots are Real and Equal");
48. r1 = r2 = (-b) / (2 \* a);
49. Console.WriteLine("**\n** First root is {0:#.##}", r1);
50. Console.WriteLine("**\n** Second root is {0:#.##}", r2);
51. Console.ReadLine();
52. **break**;
53. **case** 4: Console.WriteLine("**\n** Roots are Imaginary");
54. r1 = (-b) / (2 \* a);
55. r2 = Math.Sqrt(-d1) / (2 \* a);
56. Console.WriteLine("**\n** First root is {0:#.##} + i {1:#.##}", r1, r2);
57. Console.WriteLine("**\n** Second root is {0:#.##} - i {1:#.##}", r1, r2);
58. Console.ReadLine();
59. **break**;
60. }
61. }
62. }
64. **class** Roots
65. {
66. **public** **static** **void** Main()
67. {
68. Quadraticroots qr = new Quadraticroots();
69. qr.read();
70. qr.compute();
71. }
72. }
73. }

advertisements

Here is the output of the C# Program:

To find the roots of a quadratic equation of the form a\*x\*x + b\*x + c = 0

Enter value for a : 3.5

Enter value for b : 2.5

Enter value for c : 1.0

Roots are Imaginary

First root is -.36 + i .4

Second root is -.36 - i .4

C# Program to Display Absolute value of a Number

This C# Program Displays Absolute value of a Number. “Absolute value” means to remove any negative sign in front of a number, and to think of all numbers as positive (or zero).

Here is source code of the C# Program to Display Absolute value of a Number. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Display Absolute value of a Number*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
9. **namespace** example
10. {
11. **internal** **class** Program
12. {
13. **private** **static** **void** Main(**string**[] args)
14. {
15. **int** num;
16. Console.Write("Enter a number:");
17. num = Convert.ToInt32(Console.ReadLine());
18. **if** (num < 0)
19. {
20. num = num \* -1;
21. }
23. Console.WriteLine("Absolute value : " + num);
24. Console.ReadLine();
25. }
26. }
27. }

advertisements

Here is the output of the C# Program:

Output:

Enter a number:-50

Absolute value : 50

C# Program to Add Two Dates

This C# Program Adds Two Dates . Here the new date is found by adding using function AddDays().

Here is source code of the C# Program to Add Two Dates . The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Add Two Dates*
3. *\*/*
4. **using** System;
5. **namespace** DateAndTime
6. {
7. **class** Program
8. {
9. **static** **int** Main()
10. {
11. DateTime SDate = new DateTime(2010, 10, 7);
12. Console.WriteLine("Starting Date : {0}", SDate);
13. DateTime EDate = startDate.AddDays(10);
14. Console.WriteLine("Ending Date : {0}**\n**", EDate);
15. Console.ReadLine();
16. **return** 0;
17. }
18. }
19. }

advertisements

Here is the output of the C# Program:

Starting Date : 10/7/2010 12:00:00 AM

Ending Date : 10/17/2010 12:00:00 AM

C# Program to Generate Register Number automatically for 100 Students using Static Constructor

This C# Program Generates Register Number automatically for 100 Students using Static Constructor. Here the static constructor is used to initialize static fields of a class with specific values.

Here is source code of the C# Program to Generate Register Number automatically for 100 Students using Static Constructor. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Generate Register Number automatically for 100 Students using Static Constructor*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
8. **namespace** staticprog
9. {
10. **class** sample
11. {
12. **int** regnumber;
13. **static** **int** nextnum;
14. **static** sample()
15. {
16. nextnum=1000;
17. }
18. sample()
19. {
20. regnumber=++nextnum;
21. }
22. **public** **static** **void** Main(**string**[] args)
23. {
24. sample s=new sample();
25. Console.WriteLine("#1 : {0}",s.regnumber);
26. s=new sample();
27. Console.WriteLine("#2 : {0}",s.regnumber);
28. s = new sample();
29. Console.WriteLine("#3 : {0}", s.regnumber);
30. Console.ReadLine();
31. }
32. }
33. }

advertisements

Here is the output of the C# Program:

#1 : 1001

#2 : 1002

#3 : 1003

C# Program to Find LCM

This C# Program Finds and Display the L.C.M of a Given Number.The least common multiple (LCM) of two numbers is the smallest number that is a multiple of both.

Here is source code of the C# Program to Find and Display the L.C.M of a Given Number. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Find and Display the L.C.M of a Given Number*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
8. **namespace** ConsoleApplication9
9. {
10. **class** Program
11. {
12. **public** **static** **void** Main(**string**[] args)
13. {
14. **int** num1, num2, x, y, lcm = 0;
15. Console.Write("Enter the First Number : ");
16. num1 = **int**.Parse(Console.ReadLine());
17. Console.Write("Enter the Second Number : ");
18. num2 = **int**.Parse(Console.ReadLine());
19. x = num1;
20. y = num2;
21. **while** (num1 != num2)
22. {
23. **if** (num1 > num2)
24. {
25. num1 = num1 - num2;
26. }
27. **else**
28. {
29. num2 = num2 - num1;
30. }
31. }
32. lcm = (x \* y) / num1;
33. Console.Write("Least Common Multiple is : " + lcm);
34. Console.Read();
35. }
36. }
37. }

advertisements

Here is the output of the C# Program:

Enter the First Number : 2

Enter the Second Number : 4

Least Common Multiple : 4

#### 5. C# Examples on Permutations and Combinations and Evaluating Volume and Surface Area of Solids

Product of a given number and its quantity is called a multiple. The following programs perform addition operation on multiples of the given numbers and displays them. The programs evaluate the square root and cube root of various numbers. The others programs in the section perform permutation and combination operations on various numbers. No of arrangements made is called Permutation. The numbers of arrangements and selections is called Combination. The remaining programs evaluate the surface area and volume of different solids like Cone, Sphere and Rectangle.

C# Program to Print all the Multiples of 17 which are Less than 100

This C# Program Prints all the Multiples of 17 which are Less than 100. Here all the multiples of 17 are displayed.

Here is source code of the C# Program to Print all the Multiples of 17 which are Less than 100. The C# program is successfully compiled and executed with Microsoft Visual  
Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Print all the Multiples of 17 which are Less than 100*
3. *\*/*
4. **using** System;
5. **class** program
6. {
7. **public** **static** **void** Main()
8. {
9. **int** a,i;
10. Console.WriteLine("Multiples of 17 are : ");
11. **for** (i = 1; i < 100; i++)
12. {
13. a = i % 17;
14. **if** (a == 0)
15. {
16. Console.WriteLine(i);
17. }
18. }
19. Console.Read();
20. }
21. }

advertisements

Here is the output of the C# Program:

Multiples of 17 are :

17

34

51

68

85

C# Program to Print the Sum of all the Multiples of 3 and 5

This C# Program Prints the Sum of all the Multiples of 3 and 5. Here the multiples of 3 and 5 are found and the sum of all the multiples are calculated and are displayed.

Here is source code of the C# Program to Print the Sum of all the Multiples of 3 and 5. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\*C# Program to Print the Sum of all the Multiples of 3 and 5*
3. *\*/*
4. **using** System;
5. **class** program
6. {
7. **public** **static** **void** Main()
8. {
9. **int** a, b, i, Sum = 0;
10. **for** (i = 1; i < 100; i++)
11. {
12. a = i % 3;
13. b = i % 5;
14. **if** (a == 0 || b == 0)
15. {
16. Console.Write("{0}**\t**", i);
17. Sum = Sum + i;
18. }
19. }
20. Console.WriteLine("**\n**The Sum of all the Multiples of 3 or 5 Below 100 : {0}", Sum);
21. Console.Read();
22. }
23. }

advertisements

Here is the output of the C# Program:

3 5 6 9 10 12 15 18 20 21 24 25 27 30 33 35 36 39 40 42 45 48 50 51 54 55 57 60 63 65 66 69 70 72 75 78 80 81 84 85 87 90 93 95 96 99

The Sum of all the Multiples of 3 or 5 Below 100 : 2318

C# Program to Find the Cube Root of a Given Number

This C# Program Finds the Cube Root of a Given Number. Here the cube root of the given number is obtained using the math function.

Here is source code of the C# Program to Find the Cube Root of a Given Number. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Find the Cube Root of a Given Number*
3. *\*/*
4. **using** System;
5. **class** CubeRoot
6. {
7. **public** **static** **void** Main()
8. {
9. **double** num, res;
10. Console.Write("Enter the Number : ");
11. num = **double**.Parse(Console.ReadLine());
12. res = Math.Ceiling(Math.Pow(num, (**double**)1 / 3));
13. Console.Write("Cube Root : " + res);
15. }
16. }

advertisements

Here is the output of the C# Program:

Enter the Number : 8

Cube Root : 2

C# Program to Find Square Root of a Given Number

This C# Program Finds Square Root of a Given Number. Here the sqrt which is a math function helps in finding the squareroot of the given number.

Here is source code of the C# Program to Find Square Root of a Given Number. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Find Square Root of a Given Number*
3. *\*/*
4. **using** System;
5. **using** System.Text;
6. **using** System.Collections;
7. **using** System.Data;
8. **namespace** Cons
9. {
10. **public** **class** squareroot
11. {
12. **public** **static** **void** Main()
13. {
14. Console.WriteLine("Enter a Number : ");
15. **int** Number = Convert.ToInt16(Console.ReadLine());
16. **double** SqrtNumber = Math.Sqrt(Number);
17. Console.WriteLine("Square root of {0} is: {1}", Number, SqrtNumber);
18. Console.ReadLine();
19. }
20. }
21. }

advertisements

Here is the output of the C# Program:

Enter a Number : 36

Square Root of 36 is 6

C# Program to Calculate Area and Volume of a Cone

This C# Program Calculates Area and Volume of a Cone. Here radius and height of the cone is obtained from the user and the surface area and its volume its calculated.

Here is source code of the C# Program to Calculate Area and Volume of a Cone. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Calculate Area and Volume of a Cone*
3. *\*/*
4. **using** System;
5. **using** System.IO;
6. **class** program
7. {
8. **public** **static** **void** Main()
9. {
10. **double** r, h, surface\_area, volume;
11. **double** PI = 3.14;
12. Console.WriteLine("Enter the Radius and Height of a cone : ");
13. r = Convert.ToDouble(Console.ReadLine());
14. h = Convert.ToDouble(Console.ReadLine());
15. surface\_area = PI \* r \* (r + Math.Sqrt(r \* r + h \* h));
16. volume = (1.0 / 3) \* PI \* r \* r \* h;
17. Console.WriteLine("Surface Area of cone is : {0} ", surface\_area);
18. Console.WriteLine("Volume of Cone is : {0}", volume);
19. Console.Read();
20. }
21. }

advertisements

Here is the output of the C# Program:

Enter the Radius and Height of a cone : 3 3

Surface Area of cone is : 68.2256752726637

Volume of Cone is : 28.26

C# Program to Calculate nPr

This C# Program Calculates nPr. Here There are n! (n factorial) permutations of n symbols. A r-permutation of n symbols is a permutation of r of them. There are n!/(n – r)! different r-permutations of n symbols.

Here is source code of the C# Program to Calculate nPr. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Calculate nPr*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
9. **namespace** ConsoleApplication40
10. {
11. **class** Program
12. {
13. **static** **void** Main(**string**[] args)
14. {
15. **int** n, r, per, fact, fact1;
16. Console.WriteLine("Enter the Value of 'n' and 'r' to find the Permutation :");
17. n = Convert.ToInt32(Console.ReadLine());
18. r = Convert.ToInt32(Console.ReadLine());
19. fact = n;
20. **for** (**int** i = n - 1; i >= 1; i--)
21. {
22. fact = fact \* i;
23. }
24. **int** number;
25. number = n - r;
26. fact1 = number;
27. **for** (**int** i = number - 1; i >= 1; i--)
28. {
29. fact1 = fact1 \* i;
30. }
31. per = fact / fact1;
32. Console.WriteLine("Permutation : {0}",per);
33. Console.ReadLine();
34. }
35. }
36. }

advertisements

Here is the output of the C# Program:

Enter the value of 'n' and 'r' to find the Permutation :

10

5

Permutation : 30240

C# Program to find Volume and Surface Area of a Sphere

This C# Program finds Volume and Surface Area of a Sphere. Here radius and height of the sphere is obtained from the user and the surface area and its volume its calculated.

Here is source code of the C# Program to find Volume and Surface Area of a Sphere. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to find Volume and Surface Area of a Sphere*
3. *\*/*
4. **using** System;
5. **using** System.IO;
6. **class** program
7. {
8. **public** **static** **void** Main()
9. {
10. **double** r, surface\_area, volume;
11. **double** PI = 3.14;
12. Console.WriteLine("Enter the Radius of the Sphere: ");
13. r = Convert.ToDouble(Console.ReadLine());
14. surface\_area = 4\* PI \* r \* r;
15. volume = (4.0 / 3) \* PI \* r \* r \* r;
16. Console.WriteLine("Surface Area of Sphere is : {0} ", surface\_area);
17. Console.WriteLine("Volume of Sphere is : {0}", volume);
18. Console.Read();
19. }
20. }

advertisements

Here is the output of the C# Program:

Enter the Radius of Sphere : 5

Surface Area of Sphere is : 314

Volume of Sphere is : 523.333333333333333

C# Program to Calculate Perimeter of Circle and Rectangle

This C# Program Calculates Perimeter of Circle and Rectangle. Here length and breadth of the rectangle and radius of the circle is also obtained and perimeter of circle and rectangle is calculated.

Here is source code of the C# Program to Calculate Perimeter of Circle and Rectangle. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Calculate Perimeter of Circle and Rectangle*
3. *\*/*
4. **using** System;
5. **using** System.IO;
6. **class** program
7. {
8. **public** **static** **void** Main()
9. {
11. **double** l,b,r,per\_rect,per\_cir;
12. **double** PI = 3.14;
13. Console.WriteLine("Enter the Length and Breadth : ");
14. l = Convert.ToDouble(Console.ReadLine());
15. b = Convert.ToDouble(Console.ReadLine());
16. per\_rect = 2 \* (l + b);
17. Console.WriteLine("Enter the radius of the circle : ");
18. r = Convert.ToDouble(Console.ReadLine());
19. per\_cir = 2 \* PI \* r;
20. Console.WriteLine("Perimeter of Rectangle : {0}", per\_rect);
21. Console.WriteLine("Perimeter of Circle : {0}", per\_cir);
22. Console.Read();
23. }
24. }

advertisements

Here is the output of the C# Program:

Enter the Length and Breadth :

3

2

Enter the radius of the circle :

4

Perimeter of Rectangle : 10

Perimeter of Circle : 25.12

C# Program to Calculate the Value of nCr

This C# Program Calculates the Value of nCr. Here The Combination represented by nCr andEach r combination can be arranged in r! different ways. Then the number of r-permutations is equal to the number of r combinations times r!.

Here is source code of the C# Program to Calculate the Value of nCr. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Calculate the Value of nCr*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
8. **namespace** ConsoleApplication
9. {
10. **class** Program
11. {
12. **static** **void** Main(**string**[] args)
13. {
14. **int** n, r,per,fact,fact1,fact2;
15. Console.WriteLine("Enter the Value of 'n' and 'r' to find the Permutation :");
16. n = Convert.ToInt32(Console.ReadLine());
17. r = Convert.ToInt32(Console.ReadLine());
18. fact = n;
19. **for** (**int** i = n - 1; i >= 1; i--)
20. {
21. fact = fact \* i;
22. }
23. fact2 = r;
24. **for** (**int** i = r - 1; i >= 1; i--)
25. {
26. fact2 = fact2 \* i;
27. }
28. **int** number;
29. number = n - r;
30. fact1 = number;
31. **for** (**int** i = number - 1; i >= 1; i--)
32. {
33. fact1 = fact1 \* i;
34. }
35. fact1 = fact2 \* fact1;
36. per = fact / fact1;
37. Console.WriteLine("Combination : {0}",per);
38. Console.ReadLine();
39. }
40. }
41. }

advertisements

Here is the output of the C# Program:

Enter the value of 'n' and 'r' to find the Permutation :

10

5

Combination : 252

#### 6. C# Examples on Summation Functions

The basic data-type are int, float and double. The programs in this section calculates the maximum and minimum range of values for these data-types. They also carry out the summation operation on various different series ranging from trigonometric series like sin(x), cos(x) to many different mathematical series.

C# Program to Find the Maximum Range of Values for Decimal,Float and Double Datatype

This C# Program Finds the Maximum Range of Values for Decimal,Float and Double Datatype.the maximum range of the data types are found with the maxvalue method.

Here is source code of the C# Program to Find the Maximum Range of Values for Decimal,Float and Double Datatype. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Find the Maximum Range of Values for Decimal,Float and Double Datatype*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
8. **namespace** maxdatatype
9. {
10. **class** Program
11. {
12. **static** **void** Main(**string**[] args)
13. {
14. Console.WriteLine("The Maximum Range of the Decimal Data Type is : {0} ",**Decimal**.MaxValue);
15. Console.WriteLine("The Maximum Range of the Float Data Type is : {0} ",Single.MaxValue);
16. Console.WriteLine("The Maximum Range of the Decimal Data Type is : {0} ",**Double**.MaxValue);
17. Console.WriteLine("Exponent Form : The Maximum Range of Decimal Data Type is : {0:E}", **Decimal**.MaxValue);
18. Console.WriteLine("Exponent Form : The Maximum Range of Float Data Type is : {0:E}", Single.MaxValue);
19. Console.WriteLine("Exponent Form : The Maximum Range of Double Data Type is : {0:E}", **Double**.MaxValue);
20. Console.ReadLine();
21. }
22. }
23. }

advertisements

Here is the output of the C# Program:

The Maximum Range of the Decimal Data Type is : 7922816251464337593543950335

The Maximum Range of the Float Data Type is : 3.40282347E+38

The Maximum Range of the Double Data Type is : 1.7976931348623157E+308

Exponent Form : The Maximum Range of Decimal Data Type is : 7.922816E+028

Exponent Form : The Maximum Range of Float Data Type is : 3.402823E+038

Exponent Form : The Maximum Range of Decimal Data Type is : 1.797693E+308

C# Program to Find the Minimum Range of Values for Decimal, Float and Double Datatype

This C# Program Finds the Minimum Range of Values for Decimal,Float and Double Datatype.the maximum range of the data types are found with the minvalue method.

Here is source code of the C# Program to Find the Minimum Range of Values for Decimal,Float and Double Datatype. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Find the Minimum Range of Values for Decimal,Float and Double Datatype*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
8. **namespace** maxdatatype
9. {
10. **class** Program
11. {
12. **static** **void** Main(**string**[] args)
13. {
14. Console.WriteLine("The Minimum Range of the Decimal Data Type is : {0} ",**Decimal**.MaxValue);
15. Console.WriteLine("The Minimum Range of the Float Data Type is : {0} ",Single.MaxValue);
16. Console.WriteLine("The Minimum Range of the Decimal Data Type is : {0} ",**Double**.MaxValue);
17. Console.WriteLine("Exponent Form : The Minimum Range of Decimal Data Type is : {0:E}", **Decimal**.MaxValue);
18. Console.WriteLine("Exponent Form : The Minimum Range of Float Data Type is : {0:E}", Single.MaxValue);
19. Console.WriteLine("Exponent Form : The Minimum Range of Double Data Type is : {0:E}", **Double**.MaxValue);
20. Console.ReadLine();
21. }
22. }
23. }

advertisements

Here is the output of the C# Program:

The Minimum Range of the Decimal Data Type is : -7922816251464337593543950335

The Minimum Range of the Float Data Type is : -3.40282347E+38

The Minimum Range of the Double Data Type is : -1.7976931348623157E+308

Exponent Form : The Minimum Range of Decimal Data Type is : -7.922816E+028

Exponent Form : The Minimum Range of Float Data Type is : -3.402823E+038

Exponent Form : The MinimumRange of Decimal Data Type is : -1.797693E+308

C# Program to calculate the series sin(x)=x-x^3/3!+x^5/!-x^7/7!+……

This C# Program Finds the Value of sin(x) from the series sin(x)=x-x^3/3!+x^5/5-x^7/7!

Here is source code of the C# program to Find the Value of sin(x) from the Series. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to calculate the series sin(x)=x-x^3/3!+x^5/!-x^7/7!+......*
3. *\*/*
4. **using** System;
5. **class** sine
6. {
7. **int** deg, n;
8. **public** **void** readdata()
9. {
10. Console.WriteLine("Enter the Number of Terms:");
11. n = Convert.ToInt32(Console.ReadLine());
12. Console.WriteLine("Enter the Angle in Degrees:");
13. deg = Convert.ToInt32(Console.ReadLine());
14. }
15. **public** **void** sineseries()
16. {
17. **float** x, s = 0.0f, t;
18. x = (**float**)Math.PI \* deg / 180f;
19. s = x;
20. t = x;
21. **for** (**int** i = 1; i <= n; i++)
22. {
23. t = (-t \* x \* x) / ((2 \* i) \* (2 \* i + 1));
24. s = s + t;
25. }
26. Console.WriteLine("Sin({0})={1}", deg, s);
27. }
28. **public** **static** **void** Main()
29. {
30. sine s = new sine();
31. s.readdata();
32. s.sineseries();
33. }
34. }

advertisements

Here is the output of the C# Program:

Enter the Number of Terms:

20

Enter the Angle in Degrees:

90

Sin(90)=0.99999994

C# Program to calculate the series sin(x)=x-x^3/3!+x^5/!-x^7/7!+……

This C# Program Finds the Value of sin(x) from the series sin(x)=x-x^3/3!+x^5/5-x^7/7!

Here is source code of the C# program to Find the Value of sin(x) from the Series. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to calculate the series sin(x)=x-x^3/3!+x^5/!-x^7/7!+......*
3. *\*/*
4. **using** System;
5. **class** sine
6. {
7. **int** deg, n;
8. **public** **void** readdata()
9. {
10. Console.WriteLine("Enter the Number of Terms:");
11. n = Convert.ToInt32(Console.ReadLine());
12. Console.WriteLine("Enter the Angle in Degrees:");
13. deg = Convert.ToInt32(Console.ReadLine());
14. }
15. **public** **void** sineseries()
16. {
17. **float** x, s = 0.0f, t;
18. x = (**float**)Math.PI \* deg / 180f;
19. s = x;
20. t = x;
21. **for** (**int** i = 1; i <= n; i++)
22. {
23. t = (-t \* x \* x) / ((2 \* i) \* (2 \* i + 1));
24. s = s + t;
25. }
26. Console.WriteLine("Sin({0})={1}", deg, s);
27. }
28. **public** **static** **void** Main()
29. {
30. sine s = new sine();
31. s.readdata();
32. s.sineseries();
33. }
34. }

advertisements

Here is the output of the C# Program:

Enter the Number of Terms:

20

Enter the Angle in Degrees:

90

Sin(90)=0.99999994

#### O. C# Programming Examples on Strings

The programs in this section illustrate various operations on strings like simple operations, reversal, padding, trimming operations, replacement and conversion operations and other advanced operations on strings.

#### 1. C# Examples on Simple Operations on Strings

The following programs joins two string which is called concatenation. The other programs in the section produces the marksheet for the students of a class, prints the abbreviation for the given text, prints the date in the given string, displays the substrings that the string contains and evaluates the sum of all the digits in a string.

C# Program to Generate the Marksheet of the Student

This C# Program Generates the Marksheet of the Student. Here the student details are obtained from the user and the marksheet is generated based on the details.

Here is source code of the C# Program to Generate the Marksheet of the Student. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Generate the Marksheet of the Student*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
9. **namespace** Marksheet1
10. {
11. **class** Program
12. {
13. **static** **void** Main(**string**[] args)
14. {
15. **int** r, m1, m2, m3, t;
16. **float** p;
17. **string** n;
18. Console.WriteLine("Enter Roll Number :");
19. r = Convert.ToInt32(Console.ReadLine());
20. Console.WriteLine("Enter Student Name :");
21. n = Console.ReadLine();
22. Console.WriteLine("Mark of Subject1 : ");
23. m1 = Convert.ToInt32(Console.ReadLine());
24. Console.WriteLine("Mark of Subject2 : ");
25. m2 = Convert.ToInt32(Console.ReadLine());
26. Console.WriteLine("Mark of Subject3 : ");
27. m3 = Convert.ToInt32(Console.ReadLine());
28. t = m1 + m2 + m3;
29. p = t / 3.0f;
30. Console.WriteLine("Total : " + t);
31. Console.WriteLine("Percentage : " + p);
32. **if** (p >= 35 && p < 50)
33. {
34. Console.WriteLine("Grade is C");
35. }
36. **if** (p >= 50 && p <= 60)
37. {
38. Console.WriteLine("Grade is B");
39. }
40. **if** (p > 60 && p <= 80)
41. {
42. Console.WriteLine("Grade is A");
43. }
44. **if** (p > 80 && p <= 100)
45. {
46. Console.WriteLine("Grade is A+");
47. }
48. Console.ReadLine();
49. }
50. }
51. }

advertisements

Here is the output of the C# Program:

Enter RollNumber :

48

Enter Student Name :

sri

Mark of Subject1 :

90

Mark of Subject2 :

80

Mark of Subject3 :

70

Total : 240

Percentage : 80

Grade is A

C# Program to Display the Abbreviation of a Text

This C# Program Displays the Abbreviation of a Text. Here the string is obtained from the user and the abbreviation of the corresponding string is found and displayed.

Here is source code of the C# Program to Display the Abbreviation of a Text. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Display the Abbreviation of a Text*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
8. **class** abbreviation
9. {
10. **string** str;
11. **public** **void** readdata()
12. {
13. Console.WriteLine("Enter a String :");
14. str=Console.**In**.ReadLine();
15. }
16. **public** **void** abbre()
17. {
18. **char**[] c, result;
19. **int** j = 0;
20. c = new **char**[str.Length];
21. result = new **char**[str.Length];
22. c = str.ToCharArray();
23. result[j++] = (**char**)((**int**)c[0] ^ 32);
24. result[j++] = '.';
25. **for** (**int** i = 0; i < str.Length - 1; i++)
26. {
27. **if** (c[i] == ' ' || c[i] == '**\t**' || c[i] == '**\n**')
28. {
29. **int** k = (**int**)c[i + 1] ^ 32;
30. result[j++] = (**char**)k;
31. result[j++] = '.';
32. }
33. }
34. Console.Write("The Abbreviation for {0} is ", str);
35. Console.WriteLine(result);
36. Console.ReadLine();
38. }
39. **public** **static** **void** Main()
40. {
41. abbreviation obj=new abbreviation();
42. obj.readdata();
43. obj.abbre();
44. }
45. }

advertisements

Here is the output of the C# Program:

Enter a String :

meenakshi sundarajan engineering college

The Abbreviation for meenakshi sundarajan engineering college is M.S.E.C.

C# Program to Display Date in String

This C# Program Displays Date in String. Here the date is displayed and month, day and year are separately displayed.

Here is source code of the C# Program to Display Date in String. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Display Date in String*
3. *\*/*
4. **using** System;
5. **namespace** DateAndTime
6. {
7. **class** Program
8. {
9. **static** **int** Main()
10. {
11. DateTime date = new DateTime(2013,6, 23);
12. **string** strDate = date.ToString("M");
13. Console.WriteLine("Date and Time : {0}", date);
14. Console.WriteLine("Month and Date : {0}", strDate);
15. Console.Read();
16. **return** 0;
17. }
18. }
19. }

advertisements

Here is the output of the C# Program:

Date and Time : 6/23/2013 12:00:00 AM

Month and Date : June 23

C# Program to List all Substrings in a given String

This C# Program Lists all Substrings in a given String. Here Substring function extracts strings. It requires that you indicate a start index and a length. It then returns a completely new string with the characters in that range.

Here is source code of the C# Program to List all Substrings in a given String. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to List all Substrings in a given String*
3. *\*/*
4. **using** System;
5. **namespace** mismatch
6. {
7. **class** Program
8. {
9. **string** **value**, substring;
10. **int** j, i;
11. **string**[] a = new **string**[5];
12. **void** input()
13. {
14. Console.WriteLine("Enter the String : ");
15. **value** = Console.ReadLine();
16. Console.WriteLine("All Possible Substrings of the Given String are :");
17. **for** (i = 1; i <=**value**.Length; i++)
18. {
19. **for** (j = 0; j <= **value**.Length - i; j++)
20. {
21. substring = **value**.Substring(j, i);
22. a[j] = substring;
23. Console.WriteLine(a[j]);
24. }
25. }
26. }
27. **public** **static** **void** Main()
28. {
29. Program pg = new Program();
30. pg.input();
31. Console.ReadLine();
32. }
33. }
34. }

advertisements

Here is the output of the C# Program:

Enter the String : abab

All Possible Substrings of the Given String are :

a

b

a

b

ab

ba

ab

aba

bab

abab

C# Program to Read a String and find the Sum of all Digits in the String

This C# Program Reads a String and find the Sum of all Digits in the String. Here The program accepts a character string, then add all the character’s integer value, thereby summing all the digits of a string.

Here is source code of the C# Program to Print the Sum of all the Multiples of 3 and 5. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Read a String and find the Sum of all Digits in the String*
3. *\*/*
4. **using** System;
5. **class** program
6. {
7. **public** **static** **void** Main()
8. {
9. **char**[] string1 = new **char**[20];
10. **int** count, nc = 0, sum = 0, n, i;
11. Console.WriteLine("Enter the Length of the sentence :");
12. n = **int**.Parse(Console.ReadLine());
13. Console.WriteLine("Enter the string1 containing both digits and alphabet :");
14. **for** (i = 0; i < n; i++)
15. {
16. string1[i] = Convert.ToChar(Console.Read());
17. }
19. **for** (count = 0; string1[count] != '**\0**'; count++)
20. {
21. **if** ((string1[count] >= '0') && (string1[count] <= '9'))
22. {
23. nc += 1;
24. sum += (string1[count] - '0');
25. }
26. }
27. Console.WriteLine("NO. of Digits in the string1 = {0}", nc);
28. Console.WriteLine("Sum of all digits = {0}", sum);
29. Console.ReadLine();
30. Console.ReadLine();
31. }
32. }

advertisements

Here is the output of the C# Program:

Enter the Length of the sentence :

6

Enter the string1 containing both digits and alphabet :

SAN193

NO. of Digits in the string1 = 3

Sum of all digits = 13

C# Program to Concatenate Two Strings

This C# Program Concatenates Two Strings. Here the two strings are joined together to form a single string with a build in string function.

Here is source code of the C# Program to Concatenate Two Strings. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Concatenate Two Strings*
3. *\*/*
4. **using** System;
5. **class** Program
6. {
7. **static** **void** Main()
8. {
9. **string** s1 = "Good";
10. **string** s2 ="Morning";
11. **string** s3=**string**.Concat(s1, s2);
12. Console.WriteLine(s3);
13. Console.ReadLine();
14. }
15. }

advertisements

Here is the output of the C# Program:

GoodMorning

#### 2. C# Examples on Reversal, Padding and Trimming Operations

The programs in this section performs reversal of a string without using the reverse function. It adds characters to the end of a string which is called padding. It deletes the specified characters from the string which is called the trimming operation. The program also splits the given string into various parts which is called the splitter. The remaining program calculates the number of characters (length) of the string.

C# Program to Reverse a String without using Reverse function

This C# Program Reverses a String without using Reverse function. Here the string is reversed by changing the postion of charters from right to left one by one.

Here is source code of the C# Program to Reverse a String without using Reverse function. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Reverse a String without using Reverse function*
3. *\*/*
4. **using** System;
5. **class** Program
6. {
7. **static** **void** Main(**string**[] args)
8. {
9. **string** Str, reversestring = "";
10. **int** Length;
11. Console.Write("Enter A String : ");
12. Str = Console.ReadLine();
13. Length = Str.Length - 1;
14. **while** (Length >= 0)
15. {
16. reversestring = reversestring + Str[Length];
17. Length--;
18. }
19. Console.WriteLine("Reverse String Is {0}", reversestring);
20. Console.ReadLine();
21. }
22. }

advertisements

Here is the output of the C# Program:

Enter a String : sanfoundry

Reverse String is : yrdnuofnas

C# Program to Perform Padding in the String

This C# Program Performs Padding in the String. Here it creates a new string that consists of an original string that is padded with leading or trailing characters to a specified total length.

Here is source code of the C# Program to Perform Padding in the String. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Perform Padding in the String*
3. *\*/*
4. **using** System;
5. **namespace** padd
6. {
7. **class** Program
8. {
9. **static** **void** Main(**string**[] args)
10. {
11. **string** myString = "CSHARP";
12. **string** newString;
13. System.Console.WriteLine("String Before Padding : ");
14. System.Console.WriteLine(myString);
15. System.Console.WriteLine("String After Padding : ");
16. newString = myString.PadLeft(10, ' ');
17. newString = newString.PadRight(20, '\*');
18. System.Console.Write("[" + newString + "]");
19. Console.Read();
20. }
21. }
22. }

advertisements

Here is the output of the C# Program:

String Before Padding :

CSHARP

String After Padding :

[ CSHARP\*\*\*\*\*\*\*\*\*\*]

C# Program to Trim the Given String

This C# Program to Trim the Given String. Here spaces in the string is removed using the trim function.

Here is source code of the C# Program to Trim the Given String. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Trim the Given String*
3. *\*/*
4. **using** System;
5. **namespace** trim
6. {
7. **class** Program
8. {
9. **static** **void** Main(**string**[] args)
10. {
11. **string** myString = " CSHARP ";
12. System.Console.WriteLine("The String Before Trimming : (" + myString + ")");
13. System.Console.WriteLine("The String After Trimming : (" + myString.Trim() + ")");
14. Console.Read();
15. }
16. }
17. }

advertisements

Here is the output of the C# Program:

The String Before Trimming : ( CSHARP )

The String After Trimming : (CSHARP)

C# Program to Implement String Splitter

This C# Program Implements String Splitter. Here it Returns a string array that contains the substrings in this string that are delimited by elements of a specified string array.

Here is source code of the C# Program to Implement String Splitter. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Implement String Splitter*
3. *\*/*
4. **using** System;
5. **using** System.Text.RegularExpressions;
6. **class** Program
7. {
8. **static** **void** Main()
9. {
10. **string** sentence = "School had 40 Rooms, 500 Boys, 500 Girls and 250 Teachers";
11. **string**[] digits = Regex.Split(sentence, @"\D+");
12. **foreach** (**string** **value** **in** digits)
13. {
14. **int** number;
15. **if** (**int**.TryParse(**value**, **out** number))
16. {
17. Console.Write(**value**);
18. }
19. Console.ReadLine();
20. }
21. }
22. }

advertisements

Here is the output of the C# Program:

40

500

500

250

C# Program to Split a String Collections into Groups

This C# Program Splits a String Collections into Groups. Here the Given String is splited into Groups and are displayed.

Here is source code of the C# Program to Split a String Collections into Groups. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Split a String Collections into Groups*
3. *\*/*
4. **using** System;
5. **using** System.IO;
6. **using** System.Collections;
7. **using** System.Linq;
8. **class** program
9. {
10. **static** **void** SendEmail(**string** email)
11. {
12. Console.WriteLine(email);
13. }
14. **static** **void** Main(**string**[] args)
15. {
16. **string**[] email = {"One@aaa.com", "Two@aaa.com",
17. "Three@aaa.com", "Four@aaa.com",
18. "Five@aaa.com", "Six@aaa.com",
19. "Seven@aaa.com", "Eight@aaa.com"};
20. **var** Grp = **from** i **in** Enumerable.Range(0, email.Length)
21. **group** email[i] **by** i / 3;
22. **foreach** (**var** mail **in** Grp)
23. SendEmail(**string**.**Join**(";", mail.ToArray()));
24. Console.ReadLine();
25. }
26. }

advertisements

Here is the output of the C# Program:

One@aaa.com;Two@aaa.com;

Three@aaa.com;Four@aaa.com;

Five@aaa.com;Six@aaa.com;

Seven@aaa.com;Eight@aaa.com

C# Program to Perform Searching using Predefined Functions

This C# Program Performs Searching using Predefined Functions. Here the string is sorted using the predefined function using sort().

Here is source code of the C# Program to Perform Searching using Predefined Functions. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Perform Searching using Predefined Functions*
3. *\*/*
4. **using** System;
5. **class** linSearch
6. {
7. **public** **static** **void** Main()
8. {
9. Console.WriteLine("Enter Number of Elements you Want to Hold in the Array ? ");
10. **string** s = Console.ReadLine();
11. **int** x = Int32.Parse(s);
12. **int**[] a = new **int**[x];
13. Console.WriteLine("Enter Array Elements :");
14. **for** (**int** i = 0; i < x; i++)
15. {
16. **string** s1 = Console.ReadLine();
17. a[i] = Int32.Parse(s1);
18. }
19. Array.Sort(a);
20. Console.WriteLine("Sorted Array : ");
21. **for** (**int** i = 0; i < x; i++)
22. {
23. Console.WriteLine("{0}", a[i]);
24. }
25. Console.WriteLine("Enter the Element to be Searched : ");
26. **string** s3 = Console.ReadLine();
27. **int** x2 = Int32.Parse(s3);
28. **int** x3 = Array.BinarySearch(a, (**Object**)x2);
29. Console.WriteLine("BinarySearch: " + x3);
30. Console.WriteLine("Element {0} is {1}", x3, a[x3]);
31. Console.Read();
32. }
33. }

advertisements

Here is the output of the C# Program:

Enter Number of Elements you Want to Hold in the Array ? 5

Enter Array Elements :

2

3

1

4

5

Sorted Array :

1

2

3

4

5

Enter the Element to be Searched : 4

Binary Search : 3

Element 3 is 4

C# Program to calculate the length of the string

This C# Program calculates the length of the string. Here the Length property calculates the length of the string and displays it.

Here is source code of the C# Program to calculate the length of the string. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to calculate the length of the string*
3. *\*/*
4. **using** System;
5. **class** Program
6. {
7. **static** **void** Main()
8. {
9. **string** s1 = "Computer";
10. Console.WriteLine("The Length of the First String is : " +s1.Length);
11. **string** s2 = "";
12. Console.WriteLine("The Length of the Second String is : " +s2.Length);
13. Console.ReadLine();
14. }
15. }

advertisements

Here is the output of the C# Program:

The Length of the First String is : 8

The Length of the Second String is : 0

#### 3. C# Examples on Replacement and Conversion Operations

The programs in this section performs the replace operation where it replaces one string by another and also replaces a character by a string. The remaining program in the section converts the uppercase characters into lowercase and vice versa.

C# Program to Replace String in String

This C# Program to Replace String in String. Here It changes all occurrences of one substring into another substring.

Here is source code of the C# Program to Replace String in String. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Replace String in String*
3. *\*/*
4. **using** System;
5. **class** Program
6. {
7. **static** **void** Main()
8. {
9. **const** **string** s = "Sun Rises in the West";
10. Console.WriteLine("Sentence Before Replacing : {0} ",s);
11. **string** s1 = s.Replace("West", "East");
12. Console.WriteLine("Sentence After Replacing : {0} ",s1);
13. Console.ReadLine();
14. }
15. }

advertisements

Here is the output of the C# Program:

Sentence Before Replacing : Sun Rises in the West

Sentence After Replacing : Sun Rises in the East

C# Program to Find the Frequency of the Word ʺisʺ in a given Sentence

This C# Program Finds the Frequency of the Word ʺisʺ in a given Sentence. Here the frequency of ‘is’ in the given string is found..

Here is source code of the C# Program to Find the Frequency of the Word ʺisʺ in a given Sentence. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Find the Frequency of the Word ʺisʺ in a given Sentence*
3. *\*/*
4. **using** System;
5. **class** program
6. {
7. **public** **static** **void** Main()
8. {
9. **string** s1;
10. Console.WriteLine("Enter the String : ");
11. s1 = Console.ReadLine();
12. Console.WriteLine(counting.CountStringOccurrences(s1, "is"));
13. Console.ReadLine();
14. }
15. }
16. **public** **static** **class** counting
17. {
18. **public** **static** **int** CountStringOccurrences(**string** text, **string** pattern)
19. {
20. **int** count = 0;
21. **int** i = 0;
22. **while** ((i = text.IndexOf(pattern, i)) != -1)
23. {
24. i += pattern.Length;
25. count++;
26. }
27. **return** count;
28. }
29. }

advertisements

Here is the output of the C# Program:

Enter the String :

Rose is a Flower

1

C# Program to Convert Upper case to Lower Case

This C# Program Converts Upper case to Lower Case. Here the String is obtained in uppercase which is converted to lowercase using tolower().

Here is source code of the C# Program to Convert Upper case to Lower Case. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Convert Upper case to Lower Case*
3. *\*/*
4. **using** System;
5. **public** **class** Program
6. {
7. **public** **static** **void** Main()
8. {
9. **string** str;
10. Console.WriteLine("Enter the String in Uppercase :");
11. str = Console.ReadLine();
12. Console.WriteLine("String in LowerCase : {0}", str.ToLower());
13. Console.ReadLine();
14. }
15. }

advertisements

Here is the output of the C# Program:

Enter the String in Uppercase : SANFOUNDRY

String in Lowercase :sanfoundry

C# Program to Replace a Character with the String

This C# Program to Replace a Character with the String. Here first padding is done and then the character is replaced with the given string.

Here is source code of the C# Program to Replace a Character with the String. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Replace a Character with the String*
3. *\*/*
4. **using** System;
5. **class** Program
6. {
7. **static** **void** Main(**string**[] args)
8. {
9. **string** s = "".PadLeft(5, 'X').Replace("X", "Sanfoundry");;
10. Console.Write("The String After the Replacement :{0}",s);
11. Console.Read();
12. }
14. }

advertisements

Here is the output of the C# Program:

The String After the Replacement :SanfoundrySanfoundrySanfoundrySanfoundrySanfoundry

#### 4. C# Examples on Advanced Operations on Strings

The programs in this section perform the count operation where in it counts the number of vowels and consonants in a string and also displays the number of lines it has counted. And the remaining programs in the section perform cryptographic operations like encryption and decryption.

C# Program to Count number of Vowels and consonants from a given String

This C# Program Counts number of Vowels and consonants from a given String. Here program is used to find the number of vowels and consonants present in the given sentence. The vowels are ‘a’, ‘e’ , ‘i’, ‘o’, ‘u’, The remaining letters are consonants.

Here is source code of the C# Program to Count number of Vowels and consonants from a given String. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Count number of Vowels and consonants from a given String*
3. *\*/*
4. **using** System;
5. **class** program
6. {
7. **public** **static** **void** Main()
8. {
9. **char**[] sentence = new **char**[100];
11. **int** i, vowels = 0, consonants = 0, special = 0, n;
12. Console.WriteLine("Enter the Length of the sentence **\n**");
13. n = **int**.Parse(Console.ReadLine());
14. **for** (i = 0; i < n; i++)
15. {
16. sentence[i] = Convert.ToChar(Console.Read());
17. }
18. **for** (i = 0; sentence[i] != '**\0**'; i++)
19. {
20. **if** ((sentence[i] == 'a' || sentence[i] == 'e' || sentence[i] ==
21. 'i' || sentence[i] == 'o' || sentence[i] == 'u') ||
22. (sentence[i] == 'A' || sentence[i] == 'E' || sentence[i] ==
23. 'I' || sentence[i] == 'O' || sentence[i] == 'U'))
24. {
25. vowels = vowels + 1;
26. }
27. **else**
28. {
29. consonants = consonants + 1;
30. }
31. **if** (sentence[i] == 't' || sentence[i] == '**\0**' || sentence[i] == ' ')
32. {
33. special = special + 1;
34. }
35. }
37. consonants = consonants - special;
38. Console.WriteLine("No. of vowels {0}", vowels);
39. Console.WriteLine("No. of consonants {0}", consonants);
40. Console.ReadLine();
41. Console.ReadLine();
42. }
43. }

advertisements

Here is the output of the C# Program:

Enter the Length of the sentence

3

san

No. of vowels 1

No. of consonants 2

C# Program to Illustrate Binary Writer

This C# Program Illustrates Binary Writer. Here Write method writes a Boolean value to the stream as a one-byte value..

Here is source code of the C# Program to Illustrate Binary Writer. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Illustrate Binary Writer*
3. *\*/*
4. **using** System;
5. **using** System.IO;
6. **class** ConsoleApplication
7. {
8. **const** **string** fileName = "program.dat";
9. **static** **void** Main()
10. {
11. Write();
12. Console.WriteLine("Using Binary Writer Class the Contents are Written ");
13. }
14. **public** **static** **void** Write()
15. {
16. **using** (BinaryWriter writer = new BinaryWriter(File.Open(fileName, FileMode.Create)))
17. {
18. writer.Write(1.250F);
19. writer.Write(@"c:\Temp");
20. }
21. }
22. }

advertisements

Here is the output of the C# Program:

Using Binary Writer Class the Contents are Written

C# Program to Randomly Generate Strings

This C# Program Randomly Generates Strings. Here strings are randomly generated using the random function.

Here is source code of the C# Program to Randomly Generate Strings. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Randomly Generate Strings*
3. *\*/*
4. **using** System;
5. **using** System.IO;
6. **static** **class** Random
7. {
8. **public** **static** **string** GetRandomString()
9. {
10. **string** path = Path.GetRandomFileName();
11. path = path.Replace(".", "");
12. **return** path;
13. }
14. }
15. **class** Program
16. {
17. **static** **void** Main()
18. {
19. Console.WriteLine(Random.GetRandomString());
20. Console.WriteLine(Random.GetRandomString());
21. Console.WriteLine(Random.GetRandomString());
22. Console.Read();
23. }
24. }

advertisements

Here is the output of the C# Program:

g4jgtjvbs7hbf

jtwoj782hggjsi

3jbws63k

C# Program to Encrypt/Decrypt using Rijndael Key

This C# Program Encrypts/Decrypts using Rijndael Key.Here the namespace that is used is system.security.cryptography and the methods and properties of the rijndael method is used.

Here is source code of the C# Program to Encrypt/Decrypt using Rijndael Key. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Encrypt/Decrypt using Rijndael Key*
3. *\*/*
4. **using** System;
5. **using** System.IO;
6. **using** System.Security.Cryptography;
7. **namespace** RijndaelManage
8. {
9. **class** Rijndael
10. {
11. **public** **static** **void** Main()
12. {
13. **try**
14. {
16. **string** original = "Data For Encryption!!!!!";
17. **using** (RijndaelManaged myRijndael = new RijndaelManaged())
18. {
20. myRijndael.GenerateKey();
21. myRijndael.GenerateIV();
22. **byte**[] encrypted = EncryptStringToBytes(original, myRijndael.Key, myRijndael.IV);
23. **string** aftdecryp = DecryptStringFromBytes(encrypted, myRijndael.Key, myRijndael.IV);
24. Console.WriteLine("Original: {0}", original);
25. Console.WriteLine("After Decryption: {0}", aftdecryp);
26. }
28. }
29. **catch** (Exception e)
30. {
31. Console.WriteLine("Error: {0}", e.Message);
32. }
33. }
34. **static** **byte**[] EncryptStringToBytes(**string** plainText, **byte**[] Key, **byte**[] IV)
35. {
36. **if** (plainText == **null** || plainText.Length <= 0)
37. **throw** new ArgumentNullException("plainText");
38. **if** (Key == **null** || Key.Length <= 0)
39. **throw** new ArgumentNullException("Key");
40. **if** (IV == **null** || IV.Length <= 0)
41. **throw** new ArgumentNullException("Key");
42. **byte**[] encrypted;
43. **using** (RijndaelManaged rijAlg = new RijndaelManaged())
44. {
45. rijAlg.Key = Key;
46. rijAlg.IV = IV;
47. ICryptoTransform encryptor = rijAlg.CreateEncryptor(rijAlg.Key, rijAlg.IV);
48. **using** (MemoryStream msEncrypt = new MemoryStream())
49. {
50. **using** (CryptoStream csEncrypt = new CryptoStream(msEncrypt, encryptor, CryptoStreamMode.Write))
51. {
52. **using** (StreamWriter swEncrypt = new StreamWriter(csEncrypt))
53. {
54. swEncrypt.Write(plainText);
55. }
56. encrypted = msEncrypt.ToArray();
57. }
58. }
59. }
60. **return** encrypted;
61. }
62. **static** **string** DecryptStringFromBytes(**byte**[] cipherText, **byte**[] Key, **byte**[] IV)
63. {
64. **if** (cipherText == **null** || cipherText.Length <= 0)
65. **throw** new ArgumentNullException("cipherText");
66. **if** (Key == **null** || Key.Length <= 0)
67. **throw** new ArgumentNullException("Key");
68. **if** (IV == **null** || IV.Length <= 0)
69. **throw** new ArgumentNullException("Key");
70. **string** plaintext = **null**;
71. **using** (RijndaelManaged rijAlg = new RijndaelManaged())
72. {
73. rijAlg.Key = Key;
74. rijAlg.IV = IV;
75. ICryptoTransform decryptor = rijAlg.CreateDecryptor(rijAlg.Key, rijAlg.IV);
76. **using** (MemoryStream msDecrypt = new MemoryStream(cipherText))
77. {
78. **using** (CryptoStream csDecrypt = new CryptoStream(msDecrypt, decryptor, CryptoStreamMode.Read))
79. {
80. **using** (StreamReader srDecrypt = new StreamReader(csDecrypt))
81. {
82. plaintext = srDecrypt.ReadToEnd();
83. }
84. }
85. }
87. }
89. **return** plaintext;
90. }
91. }
92. }

advertisements

Here is the output of the C# Program:

Original : Data for Encryption!!!!!

After Decryption : Data for Encryption!!!!!

C# Program to Count Lines in a String

This C# Program Counts Lines in a String. Here the Number of Lines in the given string is calculated and are stored in count variable and are displayed.

Here is source code of the C# Program to Count Lines in a String. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Count Lines in a String*
3. *\*/*
4. **using** System;
5. **using** System.Text.RegularExpressions;
6. **class** Program
7. {
8. **static** **void** Main()
9. {
10. **long** a = countstring("This is **\n** Sanfoundry**\n** Website");
11. Console.WriteLine("Number of Lines in the String : {0}",a);
12. Console.ReadLine();
13. }
14. **static** **long** countstring(**string** s)
15. {
16. **long** count = 1;
17. **int** start = 0;
18. **while** ((start = s.IndexOf('**\n**', start)) != -1)
19. {
20. count++;
21. start++;
22. }
23. **return** count;
24. }
25. }

advertisements

Here is the output of the C# Program:

Number of Lines in the String : 3

#### P. C# Programming Examples on Networking

The programs in this section illustrate various networking concepts.

#### C# Examples on Networking Concepts

The process of linking various devices through a common communication channel is called Networking. The section covers various programs that demonstrate the concept of Networking. These include Demonstration of XML concept, implementation of UDP Protocol, printing the IP Address of a system, performing authentication of the user and establishing the relationship between the client and a server. XML is Xtensible Markup Language that defines a set of rules for encoding the documents in a human readable format. In a client-server-relationship clients are the end users which access resources from a central computer which is the server. User Datagram Protocol is a connectionless protocol that sends and receives datagrams over an IP network.

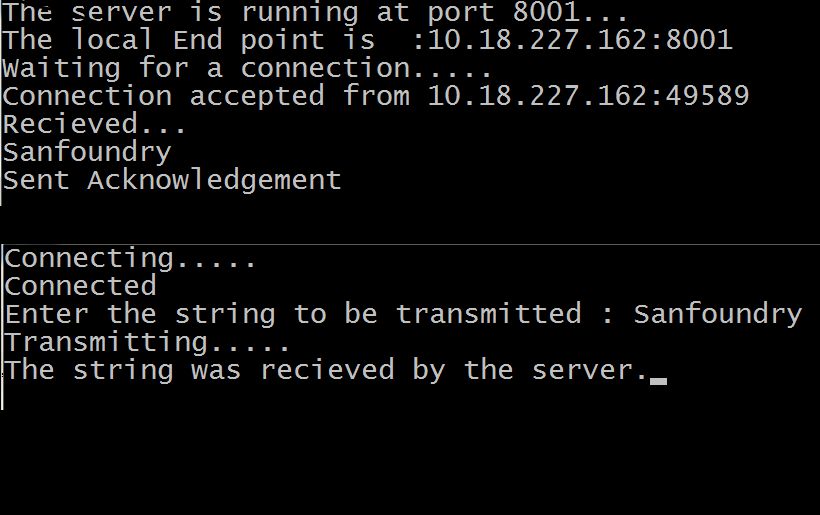
C# Program to Establish Client Server Relationship

This C# Program Establishes Client Server Relationship. Here this explains as how a relationship is established between server and a client.

Here is source code of the C# Program to Establish Client Server Relationship. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Establish Client Server Relationship*
3. *\*/*
4. *//SERVER PROGRAM*
5. **using** System;
6. **using** System.Collections.Generic;
7. **using** System.Linq;
8. **using** System.Text;
9. **using** System.Net;
10. **using** System.Net.Sockets;
11. **namespace** Server336
12. {
13. **class** Program
14. {
15. **static** **void** Main(**string**[] args)
16. {
17. **try**
18. {
19. IPAddress ipAd = IPAddress.Parse("10.18.227.162");
20. TcpListener myList = new TcpListener(ipAd, 8001);
21. myList.Start();
22. Console.WriteLine("The server is running at port 8001...");
23. Console.WriteLine("The local End point is :" + myList.LocalEndpoint);
24. Console.WriteLine("Waiting for a connection.....");
25. Socket s = myList.AcceptSocket();
26. Console.WriteLine("Connection accepted from " + s.RemoteEndPoint);
27. **byte**[] b = new **byte**[100];
28. **int** k = s.Receive(b);
29. Console.WriteLine("Recieved...");
30. **for** (**int** i = 0; i < k; i++)
31. {
32. Console.Write(Convert.ToChar(b[i]));
33. }
34. ASCIIEncoding asen = new ASCIIEncoding();
35. s.Send(asen.GetBytes("The string was recieved by the server."));
36. Console.WriteLine("**\n**Sent Acknowledgement");
37. s.Close();
38. myList.Stop();
39. }
40. **catch** (Exception e)
41. {
42. Console.WriteLine("Error..... " + e.StackTrace);
43. }
44. Console.ReadLine();
45. }
46. }
47. }
49. *//CLIENT PROGRAM*
50. **using** System;
51. **using** System.Collections.Generic;
52. **using** System.Linq;
53. **using** System.Text;
54. **using** System.IO;
55. **using** System.Net;
56. **using** System.Text;
57. **using** System.Net.Sockets;
59. **namespace** Client336
60. {
61. **class** Program
62. {
63. **static** **void** Main(**string**[] args)
64. {
65. **try**
66. {
67. TcpClient tcpclnt = new TcpClient();
68. Console.WriteLine("Connecting.....");
69. tcpclnt.Connect("10.18.227.162", 8001);
70. Console.WriteLine("Connected");
71. Console.Write("Enter the string to be transmitted : ");
72. **String** str = Console.ReadLine();
73. Stream stm = tcpclnt.GetStream();
74. ASCIIEncoding asen = new ASCIIEncoding();
75. **byte**[] ba = asen.GetBytes(str);
76. Console.WriteLine("Transmitting.....");
77. stm.Write(ba, 0, ba.Length);
78. **byte**[] bb = new **byte**[100];
79. **int** k = stm.Read(bb, 0, 100);
80. **for** (**int** i = 0; i < k; i++)
81. Console.Write(Convert.ToChar(bb[i]));
82. tcpclnt.Close();
83. Console.Read();
84. }
85. **catch** (Exception e)
86. {
87. Console.WriteLine("Error..... " + e.StackTrace);
88. }
89. }
90. }
91. }

advertisements

Here is the output of the C# Program:  


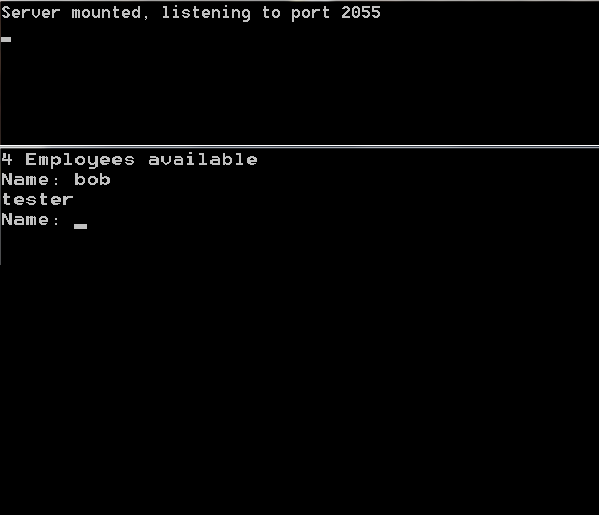
C# Program to Accept an Employee Name from Client and Sends back the Employee Job using XML

This C# Program to Accept an Employee Name from Client and Sends back the Employee Job using XML. Here name of the employee is entered in the client side window and the job the specified employee is displayed.

Here is source code of the C# Program to Accept an Employee Name from Client and Sends back the Employee Job using XML. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Accept an Employee Name from Client and Sends back the Employee*
3. *\* Job using XML*
4. *\*/*
6. *//SERVER SIDE PROGRAM*
7. **using** System;
8. **using** System.Collections.Generic;
9. **using** System.Linq;
10. **using** System.Text;
11. **using** System.Threading;
12. **using** System.IO;
13. **using** System.Net;
14. **using** System.Net.Sockets;
15. **using** System.Configuration;
16. **namespace** ServerSocket
17. {
18. **class** Program
19. {
20. **static** TcpListener listener;
21. **const** **int** LIMIT = 5;
22. **public** **static** **void** Service()
23. {
24. **while** (**true**)
25. {
26. Socket soc = listener.AcceptSocket();
27. Console.WriteLine("Connected: {0}", soc.RemoteEndPoint);
28. **try**
29. {
30. Stream s = new NetworkStream(soc);
31. StreamReader sr = new StreamReader(s);
32. StreamWriter sw = new StreamWriter(s);
33. sw.AutoFlush = **true**; *// enable automatic flushing*
34. sw.WriteLine("{0} Employees available", ConfigurationSettings.AppSettings.Count);
35. **while** (**true**)
36. {
37. **string** name = sr.ReadLine();
38. **if** (name == "" || name == **null**) **break**;
39. **string** job = ConfigurationSettings.AppSettings[name];
40. **if** (job == **null**) job = "No such employee";
41. sw.WriteLine(job);
42. }
43. s.Close();
44. }
45. **catch** (Exception e)
46. {
48. Console.WriteLine(e.Message);
49. }
50. Console.WriteLine("Disconnected: {0}", soc.RemoteEndPoint);
51. soc.Close();
52. }
53. }
54. **static** **void** Main(**string**[] args)
55. {
56. listener = new TcpListener(2055);
57. listener.Start();
59. Console.WriteLine("Server mounted, listening to port 2055");
60. **for** (**int** i = 0; i < LIMIT; i++)
61. {
62. Thread t = new Thread(new ThreadStart(Service));
63. t.Start();
64. }
65. }
66. }
67. }
68. *//XML CODING*
70. <?xml version="1.0" encoding="utf-8" ?>
71. <configuration>
72. <appSettings>
73. <**add** key = "mickey" **value**="manager"/>
74. <**add** key = "bob" **value**="tester"/>
75. <**add** key = "tom" **value**="clerk"/>
76. <**add** key = "jerry" **value**="manager"/>
77. </appSettings>
78. </configuration>
80. *//CLIENT SIDE PROGRAM*
82. **using** System;
83. **using** System.Collections.Generic;
84. **using** System.Linq;
85. **using** System.Text;
86. **using** System.IO;
87. **using** System.Net.Sockets;
88. **namespace** ClientSocket
89. {
90. **class** Program
91. {
92. **static** **void** Main(**string**[] args)
93. {
94. TcpClient client = new TcpClient("win7-PC", 2055);
95. **try**
96. {
97. Stream s = client.GetStream();
98. StreamReader sr = new StreamReader(s);
99. StreamWriter sw = new StreamWriter(s);
100. sw.AutoFlush = **true**;
101. Console.WriteLine(sr.ReadLine());
102. **while** (**true**)
103. {
104. Console.Write("Name: ");
105. **string** name = Console.ReadLine();
106. sw.WriteLine(name);
107. **if** (name == "") **break**;
108. Console.WriteLine(sr.ReadLine());
109. }
110. s.Close();
111. }
112. **finally**
113. {
114. client.Close();
115. }
117. }
118. }
119. }

advertisements

Here is the output of the C# Program:  


C# Program to Display the IP Address of the Machine

This C# Program Displays the IP Address of the Machine.It first obtains the Host Name and gets the IP address with the Host Name.

Here is source code of the C# Program to Display the IP Address of the Machine.The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Display the IP Address of the Machine*
3. *\*/*
4. **using** System;
5. **using** System.Net;
6. **namespace** Program
7. {
8. **class** Program
9. {
10. **static** **void** Main()
11. {
12. **String** strHostName = **string**.Empty; *//getting the Host Name.*
13. strHostName = Dns.GetHostName();
14. Console.WriteLine("Local Machine's Host Name: " + strHostName);
15. IPHostEntry ipEntry = Dns.GetHostEntry(strHostName);*// Using Host Name,IP address is obtained.*
16. IPAddress[] addr = ipEntry.AddressList;
18. **for** (**int** i = 0; i < addr.Length; i++)
19. {
20. Console.WriteLine("IP Address {1} : ",addr[i].ToString());
21. }
22. Console.ReadLine();
23. }
24. }
25. }

advertisements

Here is the output of the C# Program:

Local Machine's Host Name : win7-pc

IP Address : 192.168.1.2

C# Program to Illustrate how User Authentication is Done

This C# Program Illustrates how User Authentication is Done. Here The program accepts the username and password. It checks whether the password is correct with respect to the username.

Here is source code of the C# Program to Illustrate how User Authentication is Done. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Illustrate how User Authentication is Done*
3. *\*/*
4. **using** System;
5. **class** program
6. {
7. **public** **static** **void** Main()
8. {
9. **char**[] password = new **char**[10];
10. **char**[] username = new **char**[10];
11. **char** ch;
12. **int** i;
14. Console.WriteLine("Enter User name < 3 characters > : ");
15. **for** (**int** x = 0; x < 8; x++)
16. {
17. username[x] = Convert.ToChar(Console.Read());
18. }
19. Console.WriteLine("Enter the password < any 8 characters>: ");
20. **for** (i = 0; i < 8; i++)
21. {
22. ch = Convert.ToChar(Console.Read());
23. password[i] = ch;
24. ch = '\*';
25. Console.WriteLine("{0}", ch);
26. }
27. password[i] = '**\0**';
28. Console.WriteLine("**\n** Your Password is :");
29. **for** (i = 0; i < 8; i++)
30. {
31. Console.Write("{0}", password[i]);
32. }
33. Console.ReadLine();
34. Console.ReadLine();
35. }
36. }

advertisements

Here is the output of the C# Program:

Enter User name < 3 characters > :

s

r

i

Enter the password < any 8 characters>:

\*

s

\*

\*

\*

r

\*

\*

\*

i

\*

Your Password is :

s

r

i

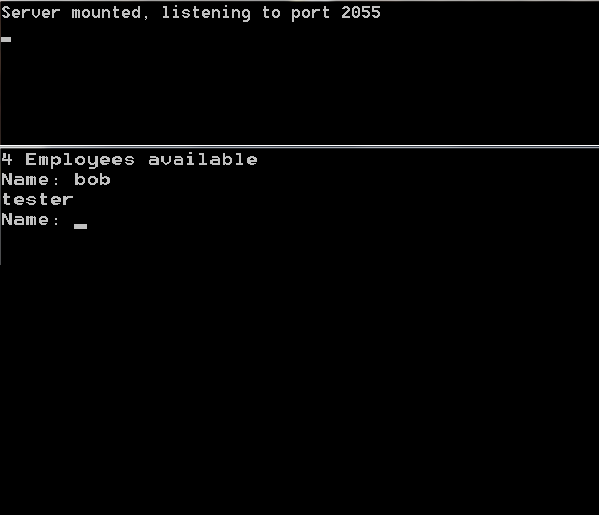
C# Program to Implement UDP

This C# Program Implements UDP. Here name of the employee is entered in the client side window and the job the specified employee is displayed.

Here is source code of the C# Program to Implement UDP. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Implement UDP*
3. *\*/*
5. *//SERVER SIDE PROGRAM*
7. **using** System;
8. **using** System.Net;
9. **using** System.Net.Sockets;
10. **using** System.Text;
11. **using** System.Configuration;
13. **class** EmployeeUDPServer
14. {
15. **public** **static** **void** Main()
16. {
17. UdpClient udpc = new UdpClient(2055);
18. Console.WriteLine("Server started, servicing on port 2055");
19. IPEndPoint ep = **null**;
20. **while** (**true**)
21. {
22. **byte**[] rdata = udpc.Receive(**ref** ep);
23. **string** name = Encoding.ASCII.GetString(rdata);
24. **string** job = ConfigurationSettings.AppSettings[name];
25. **if** (job == **null**) job = "No such employee";
26. **byte**[] sdata = Encoding.ASCII.GetBytes(job);
27. udpc.Send(sdata, sdata.Length, ep);
28. }
29. }
30. }*//XML CODING*
32. <?xml version="1.0" encoding="utf-8" ?>
33. <configuration>
34. <appSettings>
35. <**add** key = "mickey" **value**="manager"/>
36. <**add** key = "bob" **value**="tester"/>
37. <**add** key = "tom" **value**="clerk"/>
38. <**add** key = "jerry" **value**="manager"/>
39. </appSettings>
40. </configuration>
42. *//CLIENT SIDE PROGRAM*
44. **using** System;
45. **using** System.Net;
46. **using** System.Net.Sockets;
47. **using** System.Text;
48. **class** EmployeeUDPClient
49. {
50. **public** **static** **void** Main(**string**[] args)
51. {
52. UdpClient udpc = new UdpClient("Win7-PC", 2055);
53. IPEndPoint ep = **null**;
54. **while** (**true**)
55. {
56. Console.Write("Name: ");
57. **string** name = Console.ReadLine();
58. **if** (name == "") **break**;
59. **byte**[] sdata = Encoding.ASCII.GetBytes(name);
60. udpc.Send(sdata, sdata.Length);
61. **byte**[] rdata = udpc.Receive(**ref** ep);
62. **string** job = Encoding.ASCII.GetString(rdata);
63. Console.WriteLine(job);
64. }
65. }
66. }

advertisements

Here is the output of the C# Program:  


#### Q. C# Programming Examples on Matrix

The programs in this section illustrate various matrix operations.

#### C# Examples on Various Matrix Operations

A Matrix is a rectangular arrangement of rows and columns. A square matrix in which all the entries below the main diagonal are 0 is called Upper Triangular Matrix and a square matrix in which all the entries above the main diagonal are 0 is called Lower Triangular Matrix. A matrix in which all the elements are 1 is called an identity-matrix. The following programs perform numerous operations on the given set of matrices. These include Addition, Subtraction, Multiplication, Interchange, Transpose, evaluating the largest and smallest element in the matrix and demonstration of Upper Triangular Matrix and Lower Triangular Matrix.

C# Program to Perform Matrix Addition

This C# Program Performs Matrix Addition.The dimensions of the two matrices are obtained from the user and its addition is performed by adding corresponding elements.

Here is source code of the C# Program to Perform Matrix Addition. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Perform Matrix Addition*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
9. **namespace** ConsoleApplication8
10. {
11. **class** Program
12. {
13. **public** **static** **void** Main(**string**[] args)
14. {
15. **int** m, n,i,j;
16. Console.Write("Enter Number Of Rows And Columns Of Matrices A and B : ");
17. m = Convert.ToInt16(Console.ReadLine());
18. n = Convert.ToInt16(Console.ReadLine());
19. **int**[,] A = new **int**[10, 10];
20. Console.Write("**\n**Enter The First Matrix : ");
21. **for** (i = 0; i < m; i++)
22. {
23. **for** (j = 0; j < n; j++)
24. {
25. A[i, j] = Convert.ToInt16(Console.ReadLine());
26. }
27. }
29. **int**[,] B = new **int**[10, 10];
30. Console.Write("**\n**Enter The Second Matrix:");
31. **for** (i = 0; i < m; i++)
32. {
33. **for** (j = 0; j < n; j++)
34. {
35. B[i, j] = Convert.ToInt16(Console.ReadLine());
36. }
37. }
38. Console.Clear();
39. Console.WriteLine("**\n**Matrix A : ");
40. **for** (i = 0; i < m; i++)
41. {
42. **for** (j = 0; j < n; j++)
43. {
44. Console.Write(A[i, j] + "**\t**");
46. }
47. Console.WriteLine();
48. }
49. Console.WriteLine("**\n**Matrix B: ");
50. **for** (i = 0; i < m; i++)
51. {
52. **for** (j = 0; j < n; j++)
53. {
54. Console.Write(B[i, j] + "**\t**");
56. }
57. Console.WriteLine();
58. }
59. **int**[,] C = new **int**[10, 10];
60. **for** (i = 0; i < m; i++)
61. {
62. **for** (j = 0; j < n; j++)
63. {
64. C[i, j] = A[i, j] + B[i, j];
65. }
66. }
67. Console.Write("**\n**Sum Matrix :");
68. **for** (i = 0; i < m; i++)
69. {
70. **for** (j = 0; j < n; j++)
71. {
72. Console.Write(C[i, j] + "**\t**");
74. }
75. Console.WriteLine();
76. }
77. Console.Read();
78. }
79. }
80. }

advertisements

Here is the output of the C# Program:

Enter Number Of Rows And Columns Of Matrices A and B : 3 3

Enter the First Matrix :

1 2 3

2 3 4

3 4 5

Enter the Second Matrix :

1 2 3

2 1 4

1 1 5

Matrix A :

1 2 3

2 3 4

3 4 5

Matrix B :

1 2 3

2 1 4

1 1 5

Sum Matrix :

2 4 6

4 4 8

4 5 10

C# Program to Perform Matrix Subtraction

This C# Program Performs Matrix Subtraction.The dimensions of the two matrices are obtained from the user and its subtraction is performed by subtracting corresponding elements.

Here is source code of the C# program to perform Matrix Subtraction. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Perform Matrix Subtraction*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
9. **namespace** Program
10. {
11. **class** Program
12. {
13. **public** **static** **void** Main(**string**[] args)
14. {
15. **int** m, n, i, j;
16. Console.Write("Enter Number Of Rows And Columns Of Matrices A and B : ");
17. m = Convert.ToInt16(Console.ReadLine());
18. n = Convert.ToInt16(Console.ReadLine());
19. **int**[,] A = new **int**[10, 10];
20. Console.Write("**\n**Enter The First Matrix : ");
21. **for** (i = 0; i < m; i++)
22. {
23. **for** (j = 0; j < n; j++)
24. {
25. A[i, j] = Convert.ToInt16(Console.ReadLine());
26. }
27. }
29. **int**[,] B = new **int**[10, 10];
30. Console.Write("**\n**Enter The Second Matrix:");
31. **for** (i = 0; i < m; i++)
32. {
33. **for** (j = 0; j < n; j++)
34. {
35. B[i, j] = Convert.ToInt16(Console.ReadLine());
36. }
37. }
38. Console.Clear();
39. Console.WriteLine("**\n**Matrix A : ");
40. **for** (i = 0; i < m; i++)
41. {
42. **for** (j = 0; j < n; j++)
43. {
44. Console.Write(A[i, j] + "**\t**");
46. }
47. Console.WriteLine();
48. }
49. Console.WriteLine("**\n**Matrix B: ");
50. **for** (i = 0; i < m; i++)
51. {
52. **for** (j = 0; j < n; j++)
53. {
54. Console.Write(B[i, j] + "**\t**");
56. }
57. Console.WriteLine();
58. }
59. **int**[,] C = new **int**[10, 10];
60. **for** (i = 0; i < m; i++)
61. {
62. **for** (j = 0; j < n; j++)
63. {
64. C[i, j] = A[i, j] - B[i, j];
65. }
66. }
67. Console.Write("**\n**Difference Matrix :");
68. **for** (i = 0; i < m; i++)
69. {
70. **for** (j = 0; j < n; j++)
71. {
72. Console.Write(C[i, j] + "**\t**");
74. }
75. Console.WriteLine();
76. }
77. Console.Read();
78. }
79. }
80. }

advertisements

Here is the output of the C# Program:

Enter Number Of Rows And Columns Of Matrices A and B : 3 3

Enter the First Matrix :

9 8 7

6 5 4

7 8 9

Enter the Second Matrix :

6 5 5

3 4 2

1 2 3

Matrix A :

9 8 7

6 5 4

7 8 9

Matrix B :

6 5 5

3 4 2

1 2 3

Difference Matrix :

3 3 2

3 1 2

6 6 6

C# Program to Generate the Transpose of a Given Matrix

This C# Program Generates the Transpose of a Given Matrix.The transpose of a given matrix is formed by interchanging the rows and columns of a matrix..

Here is source code of the C# Program to Generate the Transpose of a Given Matrix. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Generate the Transpose of a Given Matrix*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
9. **namespace** Program
10. {
11. **class** Program
12. {
13. **public** **static** **void** Main(**string**[] args)
14. {
15. **int** m, n, i, j;
16. Console.Write("Enter the Order of the Matrix : ");
17. m = Convert.ToInt16(Console.ReadLine());
18. n = Convert.ToInt16(Console.ReadLine());
19. **int**[,] A = new **int**[10, 10];
20. Console.Write("**\n**Enter The Matrix Elements : ");
21. **for** (i = 0; i < m; i++)
22. {
23. **for** (j = 0; j < n; j++)
24. {
25. A[i, j] = Convert.ToInt16(Console.ReadLine());
26. }
27. }
28. Console.Clear();
29. Console.WriteLine("**\n**Matrix A : ");
30. **for** (i = 0; i < m; i++)
31. {
32. **for** (j = 0; j < n; j++)
33. {
34. Console.Write(A[i, j] + "**\t**");
36. }
37. Console.WriteLine();
38. }
39. Console.WriteLine("Transpose Matrix : ");
41. **for** (i = 0; i < m; i++)
42. {
43. **for** (j = 0; j < n; j++)
44. {
45. Console.Write(A[j, i] + "**\t**");
47. }
48. Console.WriteLine();
49. }
50. Console.Read();
51. }
52. }
53. }

advertisements

Here is the output of the C# Program:

Enter the Order of the Matrix : 2 2

Enter the Matrix Elements :

1 2

3 4

Matrix A :

1 2

3 4

Transpose Matrix :

1 3

2 4

C# Program to Interchange any 2 Rows of a Matrix

This C# Program Interchanges any 2 Rows of a Matrix. Here the number of rows,columns and the elements of the matrix are obtained from the user along with the rows that has to be interchanged.

Here is source code of the C# Program to Interchange any 2 Rows of a Matrix. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Interchange any 2 Rows of a Matrix*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
8. **class** interchangerow
9. {
10. **int** m, n;
11. **int**[,] a;
12. **public** interchangerow(**int** x, **int** y)
13. {
14. m = x;
15. n = y;
16. a = new **int**[m, n];
17. }
18. **public** **void** readmatrix()
19. {
20. Console.WriteLine("Enter the Elements : ");
21. **for** (**int** i = 0; i < m; i++)
22. {
23. **for** (**int** j = 0; j < n; j++)
24. {
25. Console.WriteLine("a[{0},{1}]=", i, j);
26. a[i, j] = Convert.ToInt32(Console.ReadLine());
27. }
28. }
29. }
30. **public** **void** printmax()
31. {
32. Console.WriteLine("Given Matrix : ");
33. **for** (**int** i = 0; i < m; i++)
34. {
35. **for** (**int** j = 0; j < n; j++)
36. {
37. Console.Write("{0}**\t**", a[i, j]);
39. }
40. Console.WriteLine();
41. }
42. }
43. **public** **void** interchange()
44. {
45. Console.WriteLine("Enter the Row Number to Interchange : ");
46. **int** i = Convert.ToInt32(Console.ReadLine());
47. Console.WriteLine("Enter the Row Number with which Interchange is to be Accomplished :");
48. **int** j = Convert.ToInt32(Console.ReadLine());
49. **for** (**int** k = 0; k < n; k++)
50. {
51. **int** temp = a[i - 1, k];
52. a[i - 1, k] = a[j - 1, k];
53. a[j - 1, k] = temp;
54. }
55. }
56. **public** **static** **void** Main()
57. {
58. **int** x, y;
59. interchangerow obj;
60. Console.Write("Enter the Number of Rows");
61. x = Convert.ToInt32(Console.ReadLine());
62. Console.Write("Enter the Number of Columns");
63. y = Convert.ToInt32(Console.ReadLine());
64. obj = new interchangerow(x, y);
65. obj.readmatrix();
66. obj.printmax();
67. obj.interchange();
68. obj.printmax();
69. Console.ReadLine();
70. }
71. }

advertisements

Here is the output of the C# Program:

Enter the Number of Rows : 3

Enter the Number of Columns : 3

Enter the Elements :

a[0,0]=1

a[0,1]=2

a[0,2]=3

a[1,0]=4

a[1,1]=5

a[1,2]=6

a[2,0]=7

a[2,1]=8

a[2,2]=9

Given Matrix is :

1 2 3

4 5 6

7 8 9

Enter the Row Number to Interchange : 2

Enter the Row Number with which Interchange is to be Accomplished : 3

Given Matrix is :

1 2 3

7 8 9

4 5 6

C# Program to Display Upper Triangular Matrix

This C# Program Displays Upper Triangular Matrix.A square matrix is upper triangular matrix if all its entries below the main diagonal are zero.

Here is source code of the C# Program to Display Upper Triangular Matrix. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Display Upper Triangular Matrix*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
8. **namespace** ConsoleApplication8
9. {
10. **class** Program
11. {
12. **int** x;
13. **public** **static** **void** Main(**string**[] args)
14. {
15. **int** m, n, i, j;
17. Console.Write("Enter Number Of Rows And Columns Of Matrices A and B : ");
18. m = Convert.ToInt16(Console.ReadLine());
19. n = Convert.ToInt16(Console.ReadLine());
20. **int**[,] A = new **int**[10, 10];
21. Console.Write("**\n**Enter The First Matrix : ");
22. **for** (i = 0; i < m; i++)
23. {
24. **for** (j = 0; j < n; j++)
25. {
26. A[i, j] = Convert.ToInt16(Console.ReadLine());
27. }
28. }
29. Console.Clear();
30. Console.WriteLine("**\n**Matrix A : ");
31. **for** (i = 0; i < m; i++)
32. {
33. **for** (j = 0; j < n; j++)
34. {
35. Console.Write(A[i, j] + "**\t**");
37. }
38. Console.WriteLine();
39. }
41. Console.WriteLine("**\n** Setting Zero to illustrate Upper Triangular Matrix**\n**");
42. **for** (i = 0; i < m; i++)
43. {
44. Console.Write("**\n**");
45. **for** (j = 0; j < 3; j++)
46. {
47. **if** (i <= j)
48. Console.Write(A[i, j] + "**\t**");
49. **else**
50. Console.Write("0**\t**");
51. }
52. }
53. Console.ReadLine();
54. }
55. }
56. }

advertisements

Here is the output of the C# Program:

Enter Number Of Rows And Columns Of Matrices A and B : 3 3

Enter the First Matrix :

1 2 3

2 3 4

3 4 5

Matrix A :

1 2 3

2 3 4

3 4 5

Setting Zero to illustrate Upper Triangular Matrix :

1 2 3

0 3 4

0 0 5

C# Program to Check If a Given Matrix is an Identity Matrix

This C# Program Checks If a Given Matrix is an Identity Matrix. Here An identity matrix is a square matrix, of size n × n, where the diagonal elements are all 1s, and the other elements are all 0s.

Here is source code of the C# Program to Check If a Given Matrix is an Identity Matrix. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Check If a Given Matrix is an Identity Matrix*
3. *\*/*
4. **using** System;
5. **class** pro
6. {
7. **public** **static** **void** Main()
8. {
9. Console.WriteLine("Enter the order: ");
10. **int** n = **int**.Parse(Console.ReadLine());
11. **int**[,] a = new **int**[3, 3];
12. **int** i, j;
13. Console.WriteLine("**\n** Enter the matrix**\n**");
14. **for** (i = 0; i < n; i++)
15. {
16. **for** (j = 0; j < n; j++)
17. {
18. a[i, j] = Convert.ToInt16(Console.ReadLine());
19. }
20. }
22. **for** (i = 0; i < n; i++)
23. {
24. **for** (j = 0; j < n; j++)
25. {
26. **if** ((i == j && a[i, j] == 1) || (i != j && a[i, j] == 0))
27. {
28. **goto** label;
29. }
30. **else**
31. {
32. Console.WriteLine("**\n** Not an Identity Matrix");
33. i = j = n;
34. }
35. }
37. }
38. label:
39. Console.WriteLine("Identity Matrix");
40. }
41. }

advertisements

Here is the output of the C# Program:

Enter the Order : 2

Enter the Matrix :

1

0

0

1

Identity Matrix

C# Program to Find Largest Element in a Matrix

This C# Program Finds Largest Element in a Matrix. Here the largest element in the matrix is found and displayed.

Here is source code of the C# Program to Find Largest Element in a Matrix. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Find Largest Element in a Matrix*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
8. **class** arrsampl
9. {
10. **int**[,]x;
11. arrsampl()
12. {
13. x = new **int**[,] { { 12, 21, 63 }, { 40, 15, 6 } };
14. }
15. **void** printarray()
16. {
17. Console.WriteLine("Elements in the Given Matrix : ");
18. **for** (**int** i = 0; i < 2; i++)
19. {
20. **for** (**int** j = 0; j < 3; j++)
21. {
22. Console.Write(x[i, j] + "**\t**");
23. }
24. Console.WriteLine("**\n**");
25. }
26. }
27. **int** max()
28. {
29. **int** big = x[0, 0];
30. **for** (**int** i = 0; i < 2; i++)
31. {
32. **for** (**int** j = 0; j < 3; j++)
33. {
34. **if** (big < x[i, j])
35. {
36. big = x[i, j];
37. }
38. }
39. }
40. **return** big;
41. }
42. **public** **static** **void** Main()
43. {
44. arrsampl obj = new arrsampl();
45. obj.printarray();
46. Console.WriteLine("Largest Element : {0}", obj.max());
47. Console.ReadLine();
48. }
49. }

advertisements

Here is the output of the C# Program:

Elements in the Given Matrix :

12 21 63

40 15 6

Largest Element : 63

C# Program to Perform Matrix Multiplication

This C# Program Performs Matrix Multiplication. Here The matrix multiplication is performed if the number of columns of the first matrix is equal to the number of rows of the second matrix. For loop is used to display the values in a matrix format.

Here is source code of the C# Program to Perform Matrix Multiplication. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Perform Matrix Multiplication*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
9. **namespace** matrix\_multiplication
10. {
11. **class** Program
12. {
13. **static** **void** Main(**string**[] args)
14. {
15. **int** i, j,m,n;
16. Console.WriteLine("Enter the Number of Rows and Columns : ");
17. m = Convert.ToInt32(Console.ReadLine());
18. n = Convert.ToInt32(Console.ReadLine());
19. **int**[,] a = new **int**[m, n];
20. Console.WriteLine("Enter the First Matrix");
21. **for** (i = 0; i < m; i++)
22. {
23. **for** (j = 0; j < n; j++)
24. {
25. a[i, j] = **int**.Parse(Console.ReadLine());
26. }
27. }
28. Console.WriteLine("First matrix is:");
29. **for** (i = 0; i < m; i++)
30. {
31. **for** (j = 0; j < n; j++)
32. {
33. Console.Write(a[i, j] + "**\t**");
34. }
35. Console.WriteLine();
36. }
37. **int**[,] b = new **int**[m, n];
38. Console.WriteLine("Enter the Second Matrix");
39. **for** (i = 0; i < m; i++)
40. {
41. **for** (j = 0; j < n; j++)
42. {
43. b[i, j] = **int**.Parse(Console.ReadLine());
44. }
45. }
46. Console.WriteLine("Second Matrix is :");
47. **for** (i = 0; i < 2; i++)
48. {
49. **for** (j = 0; j < 2; j++)
50. {
51. Console.Write(b[i, j] + "**\t**");
52. }
53. Console.WriteLine();
54. }
55. Console.WriteLine("Matrix Multiplication is :");
56. **int**[,] c = new **int**[m, n];
57. **for** (i = 0; i < m; i++)
58. {
59. **for** (j = 0; j < n; j++)
60. {
61. c[i, j] = 0;
62. **for** (**int** k = 0; k < 2; k++)
63. {
64. c[i, j] += a[i, k] \* b[k, j];
65. }
66. }
67. }
68. **for** (i = 0; i < m; i++)
69. {
70. **for** (j = 0; j < n; j++)
71. {
72. Console.Write(c[i, j] + "**\t**");
73. }
74. Console.WriteLine();
75. }
77. Console.ReadKey();
78. }
79. }
80. }

advertisements

Here is the output of the C# Program:

Enter the First Matrix

8

7

6

10

First Matrix is :

8 7

6 10

Enter the Second Matrix

4

3

2

1

Second Matrix is :

4 3

2 1

Matrix multiplication is :

46 31

44 28

C# Program to Find Sum of the Elements of each Row of the Given Matrix

This C# Program Finds Sum of the Elements of each Row of the Given Matrix. Here the sum of the each row of the matrix is obtained by adding the elemnts of that corresponding row.

Here is source code of the C# Program to Find Sum of the Elements of each Row of the Given Matrix. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Find Sum of the Elements of each Row*
3. *\* of the Given Matrix*
4. *\*/*
5. **using** System;
6. **using** System.Collections.Generic;
7. **using** System.Text;
8. **namespace** matrix\_row\_sum
9. {
10. **class** mat
11. {
12. **int** i, j, m, n;
13. **int**[,] a = new **int**[20, 20];
14. **public** **void** getmatrix()
15. {
16. Console.WriteLine("Enter The Number of Rows : ");
17. m = **int**.Parse(Console.ReadLine());
18. Console.WriteLine("Enter The Number of Columns : ");
19. n = **int**.Parse(Console.ReadLine());
20. Console.WriteLine("Enter the Elements");
21. **for** (i = 1; i <= m; i++)
22. {
23. **for** (j = 1; j <= n; j++)
24. {
25. a[i, j] = **int**.Parse(Console.ReadLine());
26. }
27. }
28. Console.WriteLine("Given Matrix");
29. **for** (i = 1; i <= m; i++)
30. {
31. **for** (j = 1; j <= n; j++)
32. {
33. Console.Write("**\t**{0}", a[i, j]);
34. }
35. Console.WriteLine();
36. }
37. }
38. **public** **void** row()
39. {
40. **int** r;
41. **for** (i = 1; i <= m; i++)
42. {
43. r = 0;
44. **for** (j = 1; j <= n; j++)
45. {
46. r = r + a[i, j];
47. }
48. Console.WriteLine("{0} Row Sum : {1}", i, r);
49. }
50. }
51. }
52. **class** matrowsum
53. {
54. **static** **void** Main(**string**[] args)
55. {
56. mat ma = new mat();
57. ma.getmatrix();
58. ma.row();
59. Console.ReadLine();
60. }
61. }
62. }

advertisements

Here is the output of the C# Program:

Enter the Number of Rows : 2

Enter the Number of Columns : 2

Enter the Elements :

1

2

3

4

Given Matrix :

1 2

3 4

1 Row Sum : 3

2 Row Sum : 7

C# Program to Find Sum of the Elements of each Column of the Given Matrix

This C# Program Finds Sum of the Elements of each Column of the Given Matrix. Here the sum of the each column of the matrix is obtained by adding the elements of that corresponding column.

Here is source code of the C# Program to Find Sum of the Elements of each Column of the Given Matrix. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Find Sum of the Elements of each Column*
3. *\* of the Given Matrix*
4. *\*/*
5. **using** System;
6. **using** System.Collections.Generic;
7. **using** System.Text;
8. **namespace** matrix\_col
9. {
10. **class** mat
11. {
12. **int** i, j, m, n;
13. **int**[,] a = new **int**[20, 20];
14. **int**[,] c = new **int**[20, 20];
15. **public** **void** getmatrix()
16. {
17. Console.WriteLine("Enter the Number of Rows : ");
18. m = **int**.Parse(Console.ReadLine());
19. Console.WriteLine("Enter the Number of Columns : ");
20. n = **int**.Parse(Console.ReadLine());
21. Console.WriteLine("Enter the Elements");
22. **for** (i = 1; i <= m; i++)
23. {
24. **for** (j = 1; j <= n; j++)
25. {
26. a[i, j] = **int**.Parse(Console.ReadLine());
27. }
28. }
29. Console.WriteLine("Given Matrix");
30. **for** (i = 1; i <= m; i++)
31. {
32. **for** (j = 1; j <= n; j++)
33. {
34. Console.Write("**\t**{0}", a[i, j]);
35. }
36. Console.WriteLine();
37. }
38. }
40. **public** **void** col()
41. {
42. **int** c;
43. **for** (i = 1; i <= n; i++)
44. {
45. c = 0;
46. **for** (j = 1; j <= m; j++)
47. {
48. c = c + a[j, i];
49. }
50. Console.WriteLine("{0} Column Sum : {1}", i, c);
51. }
52. }
53. }
54. **class** matsum
55. {
56. **static** **void** Main(**string**[] args)
57. {
58. mat ma = new mat();
59. ma.getmatrix();
60. ma.col();
61. Console.ReadLine();
62. }
63. }
64. }

advertisements

Here is the output of the C# Program:

Enter the Number of Rows : 2

Enter the Number of Columns : 2

Enter the Elements :

1

2

3

4

Given Matrix :

1 2

3 4

1 Column Sum : 4

2 Column Sum : 6

C# Program to Find Smallest Element in a Matrix

This C# Program Finds Smallest Element in a Matrix. Here the Smallest element in the matrix is found and displayed.

Here is source code of the C# Program to Find Smallest Element in a Matrix. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Find Smallest Element in a Matrix*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
8. **class** arrsampl
9. {
10. **int**[,]x;
11. arrsampl()
12. {
13. x = new **int**[,] { { 11, 2, 61 }, { 42, 50, 3 } };
14. }
15. **void** printarray()
16. {
17. Console.WriteLine("Elements in the Given Matrix : ");
18. **for** (**int** i = 0; i < 2; i++)
19. {
20. **for** (**int** j = 0; j < 3; j++)
21. {
22. Console.Write(x[i, j] + "**\t**");
23. }
24. Console.WriteLine("**\n**");
25. }
26. }
27. **int** max()
28. {
29. **int** small = x[0, 0];
30. **for** (**int** i = 0; i < 2; i++)
31. {
32. **for** (**int** j = 0; j < 3; j++)
33. {
34. **if** (small > x[i, j])
35. {
36. small = x[i, j];
37. }
38. }
39. }
40. **return** small;
41. }
42. **public** **static** **void** Main()
43. {
44. arrsampl obj = new arrsampl();
45. obj.printarray();
46. Console.WriteLine("Smallest Element : {0}", obj.max());
47. Console.ReadLine();
48. }
49. }

advertisements

Here is the output of the C# Program:

Elements in the Given Matrix :

11 2 61

42 50 3

Smallest Element : 2

C# Program to Find the Sum of the Values on Diagonal of the Matrix

This C# Program Finds the Sum of the Values on Diagonal of the Matrix. Here the number of rows and columns are first obtained and the sum is calculated by adding the diagonal elements.

Here is source code of the C# Program to Find the Sum of the Values on Diagonal of the Matrix. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Find the Sum of the Values on*
3. *\* Diagonal of the Matrix*
4. *\*/*
5. **using** System;
6. **using** System.Collections.Generic;
7. **using** System.Text;
8. **class** mat
9. {
10. **int** i, j, m, n;
11. **int**[,] a = new **int**[20, 20];
12. **public** **void** **get**()
13. {
14. Console.WriteLine("Enter Row Value");
15. m = **int**.Parse(Console.ReadLine());
16. Console.WriteLine("Enter Column Value");
17. n = **int**.Parse(Console.ReadLine());
18. Console.WriteLine("Enter Elements one by one");
19. **for** (i = 1; i <= m; i++)
20. {
21. **for** (j = 1; j <= n; j++)
22. {
23. a[i, j] = **int**.Parse(Console.ReadLine());
24. }
25. }
26. Console.WriteLine("Given Matrix");
27. **for** (i = 1; i <= m; i++)
28. {
29. **for** (j = 1; j <= n; j++)
30. {
31. Console.Write("**\t**{0}", a[i, j]);
32. }
33. Console.WriteLine();
34. }
35. }
36. **public** **void** diag()
37. {
38. **int** d;
39. d = 0;
40. **if** (m == n)
41. {
42. **for** (i = 1; i <= m; i++)
43. {
45. **for** (j = 1; j <= n; j++)
46. {
47. **if** (i == j)
48. {
49. d = d + a[i, j];
50. }
52. }
53. }
54. Console.WriteLine("Diagonal Sum= {0}", d);
55. }
56. **else**
57. {
58. Console.WriteLine("Can't Perform Diagonal Sum");
59. }
60. }
61. **class** matsum
62. {
63. **static** **void** Main(**string**[] args)
64. {
65. mat ma = new mat();
66. ma.**get**();
67. ma.diag();
68. Console.Read();
69. }
70. }
71. }

advertisements

Here is the output of the C# Program:

Enter Row Value : 2

Enter Column Value : 2

Enter Elements One by One :

2

2

2

2

Given Matrix :

2 2

2 2

Diagonal Sum :4

C# Program to Interchange any 2 Columns of Matrix

This C# Program Interchanges any 2 Columns of a Matrix. Here the number of rows,columns and the elements of the matrix are obtained from the user along with the columns that has to be interchanged.

Here is source code of the C# Program to Interchange any 2 Columns of a Matrix. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Interchange any 2 Columns of a Matrix*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
8. **class** interchangecol
9. {
10. **int** m, n;
11. **int**[,] a;
12. **public** interchangecol(**int** x, **int** y)
13. {
14. m = x;
15. n = y;
16. a = new **int**[m, n];
17. }
18. **public** **void** readmatrix()
19. {
20. Console.WriteLine("Enter the Elements : ");
21. **for** (**int** i = 0; i < m; i++)
22. {
23. **for** (**int** j = 0; j < n; j++)
24. {
25. Console.WriteLine("a[{0},{1}] =", i, j);
26. a[i, j] = Convert.ToInt32(Console.ReadLine());
27. }
28. }
29. }
30. **public** **void** printmax()
31. {
32. Console.WriteLine("Given Matrix : ");
33. **for** (**int** i = 0; i < m; i++)
34. {
35. **for** (**int** j = 0; j < n; j++)
36. {
37. Console.Write("{0}**\t**", a[i, j]);
38. }
39. Console.WriteLine();
40. }
41. }
42. **public** **void** interchange()
43. {
44. Console.WriteLine("Enter the Column Number to Interchange : ");
45. **int** i = Convert.ToInt32(Console.ReadLine());
46. Console.WriteLine("Enter the Column Number with which Interchange is to be Accomplished :");
47. **int** j = Convert.ToInt32(Console.ReadLine());
48. **for** (**int** k = 0; k < m; k++)
49. {
50. **int** temp = a[k, i-1];
51. a[k, i-1] = a[k, j-1];
52. a[k, j-1] = temp;
53. }
54. }
55. **public** **static** **void** Main()
56. {
57. **int** x, y;
58. interchangecol obj;
59. Console.Write("Enter the Number of Rows");
60. x = Convert.ToInt32(Console.ReadLine());
61. Console.Write("Enter the Number of Columns");
62. y = Convert.ToInt32(Console.ReadLine());
63. obj = new interchangecol(x, y);
64. obj.readmatrix();
65. obj.printmax();
66. obj.interchange();
67. obj.printmax();
68. Console.ReadLine();
69. }
70. }

advertisements

Here is the output of the C# Program:

Enter the Number of Rows : 3

Enter the Number of Columns : 3

Enter the Elements :

a[0,0]=1

a[0,1]=2

a[0,2]=3

a[1,0]=4

a[1,1]=5

a[1,2]=6

a[2,0]=7

a[2,1]=8

a[2,2]=9

Given Matrix is :

1 2 3

4 5 6

7 8 9

Enter the Column Number to Interchange : 2

Enter the Column Number with which Interchange is to be Accomplished : 3

Given Matrix is :

1 3 2

7 9 5

4 6 8

C# Program to Find the Sum of the Values on Diagonal of the Matrix

This C# Program Finds the Sum of the Values on Diagonal of the Matrix. Here the number of rows and columns are first obtained and the sum is calculated by adding the diagonal elements.

Here is source code of the C# Program to Find the Sum of the Values on Diagonal of the Matrix. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Find the Sum of the Values on*
3. *\* Diagonal of the Matrix*
4. *\*/*
5. **using** System;
6. **using** System.Collections.Generic;
7. **using** System.Text;
8. **class** mat
9. {
10. **int** i, j, m, n;
11. **int**[,] a = new **int**[20, 20];
12. **public** **void** **get**()
13. {
14. Console.WriteLine("Enter Row Value");
15. m = **int**.Parse(Console.ReadLine());
16. Console.WriteLine("Enter Column Value");
17. n = **int**.Parse(Console.ReadLine());
18. Console.WriteLine("Enter Elements one by one");
19. **for** (i = 1; i <= m; i++)
20. {
21. **for** (j = 1; j <= n; j++)
22. {
23. a[i, j] = **int**.Parse(Console.ReadLine());
24. }
25. }
26. Console.WriteLine("Given Matrix");
27. **for** (i = 1; i <= m; i++)
28. {
29. **for** (j = 1; j <= n; j++)
30. {
31. Console.Write("**\t**{0}", a[i, j]);
32. }
33. Console.WriteLine();
34. }
35. }
36. **public** **void** diag()
37. {
38. **int** d;
39. d = 0;
40. **if** (m == n)
41. {
42. **for** (i = 1; i <= m; i++)
43. {
45. **for** (j = 1; j <= n; j++)
46. {
47. **if** (i == j)
48. {
49. d = d + a[i, j];
50. }
52. }
53. }
54. Console.WriteLine("Diagonal Sum= {0}", d);
55. }
56. **else**
57. {
58. Console.WriteLine("Can't Perform Diagonal Sum");
59. }
60. }
61. **class** matsum
62. {
63. **static** **void** Main(**string**[] args)
64. {
65. mat ma = new mat();
66. ma.**get**();
67. ma.diag();
68. Console.Read();
69. }
70. }
71. }

advertisements

Here is the output of the C# Program:

Enter Row Value : 2

Enter Column Value : 2

Enter Elements One by One :

2

2

2

2

Given Matrix :

2 2

2 2

Diagonal Sum :4

C# Program to Display Lower Triangular Matrix

This C# Program Displays Lower Triangular Matrix.A square matrix is lower triangular matrix if all its entries above the main diagonal are zero.

Here is source code of the C# Program to Display Lower Triangular Matrix. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Display Lower Triangular Matrix*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
8. **namespace** ConsoleApplication8
9. {
10. **class** Program
11. {
12. **int** x;
13. **public** **static** **void** Main(**string**[] args)
14. {
15. **int** m, n, i, j;
17. Console.Write("Enter Number Of Rows And Columns Of Matrices A and B : ");
18. m = Convert.ToInt16(Console.ReadLine());
19. n = Convert.ToInt16(Console.ReadLine());
20. **int**[,] A = new **int**[10, 10];
21. Console.Write("**\n**Enter The First Matrix : ");
22. **for** (i = 0; i < m; i++)
23. {
24. **for** (j = 0; j < n; j++)
25. {
26. A[i, j] = Convert.ToInt16(Console.ReadLine());
27. }
28. }
29. Console.Clear();
30. Console.WriteLine("**\n**Matrix A : ");
31. **for** (i = 0; i < m; i++)
32. {
33. **for** (j = 0; j < n; j++)
34. {
35. Console.Write(A[i, j] + "**\t**");
37. }
38. Console.WriteLine();
39. }
41. Console.WriteLine("**\n** Setting Zero to illustrate Lower Triangular Matrix**\n**");
42. **for** (i = 0; i < m; i++)
43. {
44. Console.Write("**\n**");
45. **for** (j = 0; j < 3; j++)
46. {
47. **if** (i >= j)
48. Console.Write(A[i, j] + "**\t**");
49. **else**
50. Console.Write("0**\t**");
51. }
52. }
53. Console.ReadLine();
54. }
55. }
56. }

advertisements

Here is the output of the C# Program:

Enter Number Of Rows And Columns Of Matrices A and B : 3 3

Enter the First Matrix :

1 2 3

2 3 4

3 4 5

Matrix A :

1 2 3

2 3 4

3 4 5

Setting Zero to illustrate Lower Triangular Matrix :

1 0 0

2 3 0

3 4 5

#### R. C# Programming Examples on Interfaces

The programs in this section illustrate various types of interface implementations.

#### C# Examples on Implementation of Various Types of Interfaces

An interface contains the signatures of methods, properties or events. IDumpable Interface represents a collection of dumpable objects. Ilist Interface represents a non-generic collection of objects that can be individually accessed by index. IDictionary Interface represents a generic collection of key/value pairs. IComparable allows custom sorting of objects when implemented. IEnumerable Interface provides an enumerator, which supports a simple iteration over a non-generic collection. The programs in the following section IIlustrates handling an event declared in an interface, demonstrates IDumpable Interface, demonstrates iList Interface, demonstrates iDictionary Interface, implements IComparable Interface and implements IEnumerable interface. The remaining programs in the section demonstrates the properties of an interface and implements the transactions using interface.

C# Program to IIlustrate Handling an Event Declared in an Interface

This C# Program to IIlustrate Handling an Event Declared in an Interface. Here an event is handled with an interface.

Here is source code of the C# Program to IIlustrate Handling an Event Declared in an Interface. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to IIlustrate Handling an Event Declared in an Interface*
3. *\*/*
4. **namespace** interfaceevents
5. {
6. **using** System;
8. **public** **interface** square
9. {
10. **event** EventHandler Draw;
11. }
12. **public** **interface** rectangle
13. {
14. **event** EventHandler Draw;
15. }
16. **public** **class** Shape : square, rectangle
17. {
18. **event** EventHandler DrawEvent1;
19. **event** EventHandler DrawEvent2;
20. **object** objectLock = new **Object**();
21. **event** EventHandler square.Draw
22. {
23. **add**
24. {
25. **lock** (objectLock)
26. {
27. DrawEvent1 += **value**;
28. }
29. }
30. **remove**
31. {
32. **lock** (objectLock)
33. {
34. DrawEvent1 -= **value**;
35. }
36. }
37. }
38. **event** EventHandler rectangle.Draw
39. {
40. **add**
41. {
42. DrawEvent2 += **value**;
43. }
44. **remove**
45. {
46. DrawEvent2 -= **value**;
48. }
50. }
51. **public** **void** Draw()
52. {
53. EventHandler handler = DrawEvent1;
54. **if** (handler != **null**)
55. {
56. handler(**this**, new EventArgs());
57. }
58. Console.WriteLine("Drawing a shape.");
59. handler = DrawEvent2;
60. **if** (handler != **null**)
61. {
62. handler(**this**, new EventArgs());
63. }
64. }
65. }
66. **public** **class** classA
67. {
68. *// References the shape object as an square*
69. **public** classA(Shape shape)
70. {
71. square d = (square)shape;
72. d.Draw += new EventHandler(d\_Draw);
73. }
75. **void** d\_Draw(**object** sender, EventArgs e)
76. {
77. Console.WriteLine("ClassA receives the square event.");
78. }
79. }
80. **public** **class** classB
81. {
82. **public** classB(Shape shape)
83. {
84. rectangle d = (rectangle)shape;
85. d.Draw += new EventHandler(d\_Draw);
86. }
88. **void** d\_Draw(**object** sender, EventArgs e)
89. {
90. Console.WriteLine("ClassB receives the rectangle event.");
91. }
92. }
94. **public** **class** Program
95. {
96. **static** **void** Main(**string**[] args)
97. {
98. Shape shape = new Shape();
99. classA sub = new classA(shape);
100. classB sub2 = new classB(shape);
101. shape.Draw();
102. System.Console.WriteLine("Press any key to exit.");
103. System.Console.ReadKey();
104. }
105. }
106. }

advertisements

Here is the output of the C# Program:

ClassA receives the Square event.

Drawing a shape.

ClassB receives the Rectangle event.

Press any key to exit.

C# Program to Demonstrate IDumpable Interface

This C# Program Demonstrates IDumpable Interface.Here the Dumpable Interface is demonstrated with the use of Dump method.

Here is source code of the C# Program to Demonstrate IDumpable Interface. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Demonstrate IDumpable Interface*
3. *\*/*
4. **using** System;
5. **interface** IDumpable
6. {
7. **string** Name { **get**; **set**; }
8. **void** Dump();
9. }
11. **class** Fraction : IDumpable
12. {
13. **int** z, n;
14. **string** name;
16. **public** Fraction(**int** z, **int** n)
17. {
18. **this**.z = z; **this**.n = n;
19. }
21. **public** **string** Name
22. {
23. **get**
24. {
25. **return** name;
26. }
27. **set**
28. {
29. name = **value**;
30. }
31. }
33. **public** **void** Dump()
34. {
35. Console.WriteLine("Fraction : " + z + "/" + n);
36. }
37. }
39. **class** Person : IDumpable
40. {
41. **string** name;
42. **public** **string** address;
43. **public** **int** phone;
45. **public** Person(**string** name, **string** address, **int** phone)
46. {
47. **this**.name = name; **this**.address = address; **this**.phone = phone;
48. }
50. **public** **string** Name
51. {
52. **get** { **return** name; }
53. **set** { name = **value**; }
54. }
56. **public** **void** Dump()
57. {
58. Console.WriteLine("Person Details : {0}, {1}, {2}", name, address, phone);
59. }
60. }
62. **class** Test
63. {
65. **static** **void** Main(**string**[] arg)
66. {
67. IDumpable[] a =
68. {
69. new Fraction(10,3),
70. new Fraction(9,4),
71. new Person("Tom", "INDIA", 99556677),
72. new Person("Jerry", "INDIA", 998979899),
73. };
74. a[0].Name = "f1";
75. a[1].Name = "f2";
76. **foreach** (IDumpable obj **in** a)
77. {
78. Console.Write(obj.Name + ": ");
79. obj.Dump();
80. }
81. Console.ReadLine();
82. }
84. }

advertisements

Here is the output of the C# Program:

f1 : Fraction : 10/3

f2 : Fraction : 9/4

Tom : Person Details : Tom, INDIA, 99556677

Jerry : Person Details :Jerry, INDIA, 998979899

C# Program to Demonstrate Properties of the Interface

This C# Program Demonstrates Properties of the Interface. Here the syntax for a property type on an interface declaration is different and the interface declarations do not include modifiers such as “public.”

Here is source code of the C# Program to Demonstrate Properties of the Interface. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Demonstrate Properties of the Interface*
3. *\*/*
4. **using** System;
6. **interface** IValue
7. {
8. **int** Count { **get**; **set**; }
9. **string** Name { **get**; **set**; }
10. }
12. **class** Image : IValue
13. {
14. **public** **int** Count
15. {
16. **get**;
17. **set**;
18. }
20. **string** \_name;
22. **public** **string** Name
23. {
24. **get** { **return** **this**.\_name; }
25. **set** { **this**.\_name = **value**; }
26. }
27. }
29. **class** Article : IValue
30. {
31. **public** **int** Count
32. {
33. **get**;
34. **set**;
35. }
37. **string** \_name;
39. **public** **string** Name
40. {
41. **get** { **return** **this**.\_name; }
42. **set** { **this**.\_name = **value**.ToUpper(); }
43. }
44. }
46. **class** Program
47. {
48. **static** **void** Main()
49. {
50. IValue value1 = new Image();
51. IValue value2 = new Article();
53. value1.Count++;
54. value2.Count++;
56. value1.Name = "Tom";
57. value2.Name = "Jerry";
59. Console.WriteLine(value1.Name);
60. Console.WriteLine(value2.Name);
61. Console.ReadLine();
63. }
64. }

advertisements

Here is the output of the C# Program:

Tom

JERRY

C# Program to Demonstrate iList Interface

This C# Program Demonstrates iList Interface. Here Lists and arrays implement IList and this interface is an abstraction that allows list types to be used with through a single reference type. With it, we can create a single method to receive an int[] or a List.

Here is source code of the C# Program to Demonstrate iList Interface. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Demonstrate iList Interface*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
7. **class** Program
8. {
9. **static** **void** Main()
10. {
11. **int**[] a = new **int**[3];
12. a[0] = 1;
13. a[1] = 2;
14. a[2] = 3;
15. Display(a);
17. List<**int**> list = new List<**int**>();
18. list.**Add**(5);
19. list.**Add**(7);
20. list.**Add**(9);
21. Display(list);
22. Console.ReadLine();
23. }
25. **static** **void** Display(IList<**int**> list)
26. {
27. Console.WriteLine("Count: {0}", list.Count);
28. **foreach** (**int** num **in** list)
29. {
30. Console.WriteLine(num);
31. }
32. }
33. }

advertisements

Here is the output of the C# Program:

Count: 3

1

2

3

Count: 3

5

7

9

C# Program to Demonstrate iDictionary Interface

This C# Program Demonstrates iDictionary Interface. Here This program uses both the Dictionary and SortedDictionary types. Suppose that you want to add some functionality that can work on an instance of Dictionary or an instance of SortedDictionary.

Here is source code of the C# Program to Demonstrate iDictionary Interface. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Demonstrate iDictionary Interface*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
7. **class** Program
8. {
9. **static** **void** Main()
10. {
11. Dictionary<**string**, **string**> dict = new Dictionary<**string**, **string**>();
12. dict["Tom"] = "Bob";
13. WriteKeyA(dict);
14. SortedDictionary<**string**, **string**> sort = new SortedDictionary<**string**, **string**>();
15. sort["Tom"] = "Jerry";
16. WriteKeyA(sort);
17. Console.ReadLine();
18. }
20. **static** **void** WriteKeyA(IDictionary<**string**, **string**> i)
21. {
23. Console.WriteLine(i["Tom"]);
24. }
25. }

Here is the output of the C# Program:

Bob

Jerry

C# Program to Demonstrate Transactions using Interface

This C# Program Demonstrates Transactions using Interface. Here Interfaces define properties, methods and events, which are the members of the interface and contains only the declaration of the members and has the responsibility of the deriving class to define the members.

Here is source code of the C# Program to Demonstrate Transactions using Interface. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Demonstrate Transactions using Interface*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
9. **namespace** sample
10. {
12. **public** **interface** ITransactions
13. {
14. **void** showTransaction();
15. **double** getamnt();
16. }
17. **public** **class** Transaction : ITransactions
18. {
19. **private** **string** Code;
20. **private** **string** date;
21. **private** **double** amnt;
22. **public** Transaction()
23. {
24. Code = " ";
25. date = " ";
26. amnt = 0.0;
27. }
28. **public** Transaction(**string** c, **string** d, **double** a)
29. {
30. Code = c;
31. date = d;
32. amnt = a;
33. }
34. **public** **double** getamnt()
35. {
36. **return** amnt;
37. }
38. **public** **void** showTransaction()
39. {
40. Console.WriteLine("Transaction: {0}", Code);
41. Console.WriteLine("Date: {0}", date);
42. Console.WriteLine("amnt: {0}", getamnt());
44. }
46. }
47. **class** example
48. {
49. **static** **void** Main(**string**[] args)
50. {
51. Transaction t1 = new Transaction("001", "24/06/2014", 87900.00);
52. Transaction t2 = new Transaction("002", "25/06/2014", 51900.00);
53. t1.showTransaction();
54. t2.showTransaction();
55. Console.ReadKey();
56. }
57. }
58. }

advertisements

Here is the output of the C# Program:

Transaction: 001

Date: 24/06/2014

amnt: 87900

Transaction: 002

Date: 25/06/2014

amnt: 51900

C# Program to Implement IComparable Interface

This C# Program Implements IComparable Interface.IComparable allows custom sorting of objects when implemented. When a class implements this interface, we must add the public method CompareTo(T).

Here is source code of the C# Program to Implement IComparable Interface. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Implement IComparable Interface*
3. *\*/*
4. **using** System;
5. **class** Fraction : IComparable
6. {
7. **int** z, n;
9. **public** Fraction(**int** z, **int** n)
10. {
11. **this**.z = z; **this**.n = n;
12. }
14. **public** **static** Fraction **operator** +(Fraction a, Fraction b)
15. {
16. **return** new Fraction(a.z \* b.n + a.n \* b.z, a.n \* b.n);
17. }
19. **public** **static** Fraction **operator** \*(Fraction a, Fraction b)
20. {
21. **return** new Fraction(a.z \* b.z, a.n \* b.n);
22. }
24. **public** **int** CompareTo(**object** obj)
25. {
26. Fraction f = (Fraction)obj;
27. **if** ((**float**)z / n < (**float**)f.z / f.n)
28. **return** -1;
29. **else** **if** ((**float**)z / n > (**float**)f.z / f.n)
30. **return** 1;
31. **else** **return** 0;
32. }
34. **public** **override** **string** ToString()
35. {
36. **return** z + "/" + n;
37. }
38. }
40. **class** Test
41. {
43. **static** **void** Main(**string**[] arg)
44. {
45. Fraction[] a = {
46. new Fraction(5,2),
47. new Fraction(29,6),
48. new Fraction(4,5),
49. new Fraction(10,8),
50. new Fraction(34,7)
51. };
52. Array.Sort(a);
53. Console.WriteLine("Implementing the IComparable Interface in Displaying Fractions : ");
54. **foreach** (Fraction f **in** a) Console.WriteLine(f + " ");
55. Console.WriteLine();
56. Console.ReadLine();
57. }
59. }

advertisements

Here is the output of the C# Program:

Implementing the IComparable Interface in Displaying Fractions :

4/5

10/8

5/2

29/6

34/7

C# Program to Implement IEnumerable Interface using LINQ

This C# Program Implements IEnumerable Interface using LINQ. Here it exposes an enumerator, which supports a simple iteration over a non-generic collection.

Here is source code of the C# Program to Implement IEnumerable Interface using LINQ. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Implement IEnumerable Interface using LINQ*
3. *\*/*
4. **using** System;
5. **using** System.IO;
6. **using** System.Collections;
7. **using** System.Linq;
8. **class** program
9. {
10. **public** **static** **void** Main(**string**[] args)
11. {
12. **var** t = typeof(IEnumerable);
13. **var** typesIEnum = AppDomain.CurrentDomain.GetAssemblies().SelectMany(x => x.GetTypes()).**Where**(x => t.IsAssignableFrom(x));
14. **foreach** (**var** types **in** typesIEnum)
15. {
16. Console.WriteLine(types.FullName);
17. }
18. Console.ReadLine();
19. }
20. }

advertisements

Here is the output of the C# Program:

System.Linq.Parallel.IndexedSelectQueryOperator`2

System.Linq.Parallel.IndexedSelectQueryOperator`2+IndexedSelectQueryOperatorResults

System.Linq.Parallel.IndexedWhereQueryOperator`1

System.Linq.Parallel.LastQueryOperator`1

System.Linq.Parallel.ReverseQueryOperator`1

System.Linq.Parallel.ReverseQueryOperator`1+ReverseQueryOperatorResults

System.Linq.Parallel.SelectManyQueryOperator`3

System.Linq.Parallel.SelectQueryOperator`2

System.Linq.Parallel.SelectQueryOperator`2+SelectQueryOperatorResults

System.Linq.Parallel.SingleQueryOperator`1

System.Linq.Parallel.SortQueryOperator`2

System.Linq.Parallel.SortQueryOperatorResults`2

System.Linq.Parallel.TakeOrSkipQueryOperator`1

System.Linq.Parallel.TakeOrSkipQueryOperator`1+TakeOrSkipQueryOperatorResults

System.Linq.Parallel.TakeOrSkipWhileQueryOperator`1

System.Linq.Parallel.WhereQueryOperator`1

System.Linq.Parallel.ListChunk`1

System.Linq.Parallel.Lookup`2

#### S. C# Programming Examples on Threads

The programs in this section illustrate various types of methods that are implemented on thread and the basics of thread.

#### 1. C# Examples on Basics of Thread

A thread is a path in the execution of a process. It contains a Program counter, a thread ID, a stack and a set of registers. Thread Pool is a group of threads that are waiting ready to be given work. Killing a thread means forcing the referenced thread to terminate. sleep() method is used to pause a running thread. The following programs demonstrate the creation of a thread, thread pool, pausing the execution of a running thread, illustrating a thread lock and killing a thread.

C# Program to Create a Simple Thread

This C# Program Creates a Simple Thread. Here create a new ThreadStart delegate. The delegate points to a method that will be executed by the new thread. Pass this delegate as a parameter when creating a new Thread instance. Finally, call the Thread.Start method to run the method.

Here is source code of the C# Program to Create a Simple Thread. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Create a Simple Thread*
3. *\*/*
4. **using** System;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Text;
8. **using** System.Threading;
9. **class** program
10. {
11. **public** **void** WorkThreadFunction()
12. {
13. **for** (**int** i = 0; i < 5; i++)
14. {
15. Console.WriteLine("Simple Thread");
16. }
17. }
18. }
19. **class** threprog
20. {
21. **public** **static** **void** Main()
22. {
23. program pg = new program();
24. Thread thread = new Thread(new ThreadStart(pg.WorkThreadFunction));
25. thread.Start();
26. Console.Read();
27. }
28. }

advertisements

Here is the output of the C# Program:

Simple Thread

Simple Thread

Simple Thread

Simple Thread

Simple Thread

C# Program to Create Thread Pools

This C# Program Creates Thread Pools. Here it comprises two separate tasks called Task1 and Task2 that do the simple job of outputting a message to the console in a loop. A ThreadPool class can be employed to start these two tasks without setting the properties of threads.

Here is source code of the C# Program to Create Thread Pools. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Create Thread Pools*
3. *\*/*
4. **using** System;
5. **using** System.Threading;
6. **class** ThreadPoolDemo
7. {
8. **public** **void** task1(**object** obj)
9. {
10. **for** (**int** i = 0; i <= 2; i++)
11. {
12. Console.WriteLine("Task 1 is being executed");
13. }
14. }
15. **public** **void** task2(**object** obj)
16. {
17. **for** (**int** i = 0; i <= 2; i++)
18. {
19. Console.WriteLine("Task 2 is being executed");
20. }
21. }
23. **static** **void** Main()
24. {
25. ThreadPoolDemo tpd = new ThreadPoolDemo();
26. **for** (**int** i = 0; i < 2; i++)
27. {
28. ThreadPool.QueueUserWorkItem(new WaitCallback(tpd.task1));
29. ThreadPool.QueueUserWorkItem(new WaitCallback(tpd.task2));
30. }
32. Console.Read();
33. }
34. }

advertisements

Here is the output of the C# Program:

Task 1 is being executed

Task 1 is being executed

Task 1 is being executed

Task 1 is being executed

Task 1 is being executed

Task 1 is being executed

Task 2 is being executed

Task 2 is being executed

Task 2 is being executed

Task 2 is being executed

Task 2 is being executed

Task 2 is being executed

C# Program to Pause a Thread

This C# Program Pauses a Thread. Here Sleep pauses programs. It receives a value indicating the number of milliseconds to wait.

Here is source code of the C# Program to Pause a Thread. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Pause a Thread*
3. *\*/*
4. **using** System;
5. **using** System.Threading;
6. **class** Example
7. {
8. **static** **void** Main()
9. {
10. **for** (**int** i = 0; i < 5; i++)
11. {
12. Console.WriteLine("Sleep for 2 Seconds");
13. Thread.Sleep(2000);
14. }
16. Console.WriteLine("Main thread Exits");
17. Console.ReadLine();
18. }
19. }

advertisements

Here is the output of the C# Program:

Sleep for 2 Seconds

Sleep for 2 Seconds

Sleep for 2 Seconds

Sleep for 2 Seconds

Sleep for 2 Seconds

Sleep for 2 Seconds

Main thread Exits

C# Program to Demonstrate Lock in Thread

This C# Program Demonstrates Lock in Thread. The lock keyword marks a statement block as a critical section by obtaining the mutual-exclusion lock for a given object, executing a statement, and then releasing the lock. The following example includes a lock statement..

Here is source code of the C# Program to Demonstrate Lock in Thread. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Demonstrate Lock in Thread*
3. *\*/*
5. **using** System;
6. **using** System.Threading;
8. **class** Program
9. {
10. **static** **readonly** **object** \_object = new **object**();
12. **static** **void** TEST()
13. {
14. **lock** (\_object)
15. {
16. Thread.Sleep(100);
17. Console.WriteLine(Environment.TickCount);
18. }
19. }
20. **static** **void** Main()
21. {
22. **for** (**int** i = 0; i < 10; i++)
23. {
24. ThreadStart start = new ThreadStart(TEST);
25. new Thread(start).Start();
26. }
27. }
28. }

advertisements

Here is the output of the C# Program:

900500

900593

900687

900796

900890

900999

901092

901186

901295

901389

C# Program to Kill a Thread

This C# Program Kills a Thread. Here the working of the thread is stopped when a key is pressed else the work continues.

Here is source code of the C# Program to Kill a Thread. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Kill a Thread*
3. *\*/*
4. **using** System;
5. **using** System.Threading.Tasks;
6. **using** System.Threading;
7. **class** Program
8. {
9. **static** **void** Main(**string**[] args)
10. {
12. ThreadingClass th = new ThreadingClass();
13. Thread thread1 = new Thread(th.DoStuff);
14. thread1.Start();
15. Console.WriteLine("Press any key to exit!!!");
16. Console.ReadKey();
17. th.Stop();
18. thread1.**Join**();
19. }
20. }
21. **public** **class** ThreadingClass
22. {
23. **private** **bool** flag = **false**;
24. **public** **void** DoStuff()
25. {
26. **while** (!flag)
27. {
28. Console.WriteLine(" Thread is Still Working");
29. Thread.Sleep(1000);
30. }
31. }
32. **public** **void** Stop()
33. {
34. flag = **true**;
35. }
36. }

advertisements

Here is the output of the C# Program:

Press any key to exit!!!

Thread is Still Working

Thread is Still Working

Thread is Still Working

Thread is Still Working

Thread is Still Working

Thread is Still Working

Thread is Still Working

Thread is Still Working

Thread is Still Working

#### 2. C# Examples on the Methods that are Implemented on Thread

The following programs gives the name of the thread that is currently executing, displays the thread priority – It determines which thread is to be called next for execution. Name property of the thread gets or sets the name of the thread. The sleep() method of the thread makes the thread to sleep for a specified period of time. The following programs displays the context id of a thread, demonstrates the name property and sleep method of a thread and obtains the status of a thread. It determines if the thread is in new, runnable, waiting, blocked and terminated state.

C# Program to Display the Name of the Current Thread

This C# Program Displays the Name of the Current Thread. Here a Thread is the representation of the currently running thread and the name of the current thread is found and displayed.

Here is source code of the C# Program to Display the Name of the Current Thread. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Display the Name of the Current Thread*
3. *\*/*
4. **using** System;
5. **using** System.Threading;
7. **namespace** threading
8. {
9. **class** Program
10. {
11. **static** **void** Main(**string**[] args)
12. {
13. Console.WriteLine("Current information");
14. Thread t = Thread.CurrentThread;
15. t.Name = "primarythread";
17. Console.WriteLine("Thread Name: {0}", t.Name);
18. Console.WriteLine("Thread Status: {0}", t.IsAlive);
20. Console.ReadKey();
21. }
23. }
24. }

advertisements

Here is the output of the C# Program:

Current information

Thread Name: primarythread

Thread Status: True

C# Program to Illustrate the Concept of Passing Parameter for Thread

This C# Program Illustrates the Concept of Passing Parameter for Thread. HereThe following code example shows the syntax for creating and using a ParameterizedThreadStart delegate with a static method and an instance method.

Here is source code of the C# Program to Illustrate the Concept of Passing Parameter for Thread. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Illustrate the Concept of Passing Parameter for Thread*
3. *\*/*
4. **using** System;
5. **using** System.Threading;
6. **public** **class** pgm
7. {
8. **public** **static** **void** Main()
9. {
10. Thread newThread = new Thread(pgm.work1);
11. newThread.Start(20);
12. pgm p = new pgm();
13. newThread = new Thread(p.work2);
14. newThread.Start("Instance");
15. Console.ReadLine();
16. }
17. **public** **static** **void** work1(**object** data)
18. {
19. Console.WriteLine("Static Thread Procedure : Data ='{0}'",data);
20. }
21. **public** **void** work2(**object** data)
22. {
23. Console.WriteLine("Instance Thread Procedure : Data ='{0}'", data);
24. }
25. }

advertisements

Here is the output of the C# Program:

Static Thread Procedure : Data = '20'

Instance Thread Procedure : Data = 'Instance'

C# Program to Display the Priority of the Thread

This C# Program Displays the Priority of the Thread.There are various priorities for the thread and Zero,Below Normal,Normal,Above Normal,Highest are one such.The priority of the current working thread is found and displayed.

Here is source code of the C# Program to Display the Priority of the Thread. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Display the Priority of the Thread*
3. *\*/*
4. **using** System;
5. **using** System.Threading;
7. **namespace** threading
8. {
9. **class** Program
10. {
11. **static** **void** Main(**string**[] args)
12. {
13. Console.WriteLine("\*\*\*\*\*\*\*\*\*\*Current Thread Informations\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*n");
14. Thread t = Thread.CurrentThread;
15. t.Name = "Primary\_Thread";
17. Console.WriteLine("Thread Name: {0}", t.Name);
18. Console.WriteLine("Thread Status: {0}", t.IsAlive);
19. Console.WriteLine("Priority: {0}", t.Priority);
21. Console.ReadKey();
22. }
24. }
25. }

advertisements

Here is the output of the C# Program:

Thread Name: PrimaryThread

Thread Status: True

Priority: Normal

C# Program to Find the Current Context id of the Thread

This C# Program Finds the Current Context id of the Thread. Here A Context is something associated with ContextBoundObject-objects and multiple threads can share the same context (Thread.CurrentContext).

Here is source code of the C# Program to Find the Current Context id of the Thread. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Find the Current Context id of the Thread*
3. *\*/*
4. **using** System;
5. **using** System.Threading;
7. **namespace** threading
8. {
9. **class** Program
10. {
11. **static** **void** Main(**string**[] args)
12. {
13. Console.WriteLine("\*\*\*\*\*\*\*\*\*\*Current Thread Informations\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*n");
14. Thread t = Thread.CurrentThread;
15. t.Name = "Primary\_Thread";
17. Console.WriteLine("Thread Name: {0}", t.Name);
18. Console.WriteLine("Thread Status: {0}", t.IsAlive);
19. Console.WriteLine("Priority: {0}", t.Priority);
21. Console.ReadKey();
22. }
24. }
25. }

advertisements

Here is the output of the C# Program:

Thread Name: PrimaryThread

Thread Status: True

Context ID: 0

C# Program to Assign Name to Thread by using Name Property

This C# Program Assigns Name to Thread by using Name Property. Here the Name property Gets or sets the name of the thread.

Here is source code of the C# Program to Assign Name to Thread by using Name Property. The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Assign Name to Thread by using Name Property*
3. *\*/*
4. **using** System;
5. **using** System.Threading;
6. **class** Name
7. {
8. **static** **void** Main()
9. {
10. **if** (Thread.CurrentThread.Name == **null**)
11. {
12. Thread.CurrentThread.Name = "Main";
13. Console.Write("Thread has been Named ");
14. }
15. **else**
16. {
17. Console.WriteLine("Unable to name a previously " +
18. "named thread.");
19. }
20. Console.ReadLine();
21. }
22. }

advertisements

Here is the output of the C# Program:

Thread has been Named

C# Program to Implement Sleep method of Thread

This C# Program Implements Sleep method of Thread. Here Sleep pauses programs. It receives a value indicating the number of milliseconds to wait.

Here is source code of the C# Program to Implement Sleep method of Thread. The C# program is successfully compiled and executed with Microsoft Visual  
Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Implement Sleep method of Thread*
3. *\*/*
4. **using** System;
5. **using** System.Diagnostics;
6. **using** System.Threading;
7. **class** Program
8. {
9. **static** **void** Main()
10. {
11. **var** stopwatch = Stopwatch.StartNew();
12. Thread.Sleep(500);
13. stopwatch.Stop();
14. Console.WriteLine("Elapsed Milliseconds : {0}",stopwatch.ElapsedMilliseconds);
15. Console.WriteLine("Elapsed Ticks : {0}", stopwatch.ElapsedTicks);
16. Console.WriteLine("Present Date and Time : {0}",DateTime.Now.ToLongTimeString());
17. Console.ReadLine();
18. }
19. }

advertisements

Here is the output of the C# Program:

Elapsed Milliseconds : 498

Elapsed Ticks : 1231409

Present Date and Time : 8:36:06 PM

C# Program to Obtain Status of the Current Thread

This C# Program Obtains the Status of the Current Thread. Here a Thread is the representation of the currently running thread and the true or false is displayed as a result of the status method.

Here is source code of the C# Program to Obtain Status of the Current Thread . The C# program is successfully compiled and executed with Microsoft Visual Studio. The program output is also shown below.

1. */\**
2. *\* C# Program to Obtain Status of the Current Thread*
3. *\*/*
4. **using** System;
5. **using** System.Threading;
7. **namespace** threading
8. {
9. **class** Program
10. {
11. **static** **void** Main(**string**[] args)
12. {
13. Console.WriteLine("Current information");
14. Thread t = Thread.CurrentThread;
15. t.Name = "primarythread";
17. Console.WriteLine("Thread Name: {0}", t.Name);
18. Console.WriteLine("Thread Status: {0}", t.IsAlive);
20. Console.ReadKey();
21. }
23. }
24. }

advertisements

Here is the output of the C# Program:

Current information

Thread Name: primarythread

Thread Status: True